A GMM-STRAIGHT Approach to Voice Conversion

EE 225D Project, May 2009

Stephen Shum
SID: 18066044
sshum@berkeley.edu

Abstract

This paper explores the topic of voice conversion as explored in a joint project with Percy Liang (EECS, Berkeley). For our purposes, voice conversion is the process of modifying the speech signal of one speaker (source) such that it sounds as though it had been pronounced by a different speaker (target). Following the Source-Filter model of speech production, we begin by assuming that most of a speaker's characteristics can be summarized in the spectral envelope as represented by a set of Linear Predictive Coefficients. By using a Gaussian mixture model (GMM) to model the features of the source speaker, we can then learn a mapping of features from the source to the target, and then resynthesize via various methods. In this paper, we explore different approaches to model the features and describe our results from experimenting with the resynthesis process, including the integration of a STRAIGHT vocoder system, which provides an advanced model of the excitation signal. Further discussion includes ways to immediately improve the system and how we would like to proceed in the future.

Contents

1 The Motivation

Speech is used as a way of conveying a wide range of information. Though a primary interest in human speech is to communicate the meaning of a message (a topic of much interest in Automatic Speech Recognition and Natural Language Processing), also present in the signal is secondary information that includes a speaker's identity, emotion, age, and even possible pathology. The specific correlates of these characteristics to the actual signal are still being explored in many areas of research. For now, what we do know is that the individuality amongst voices is what makes life interesting.

Voice modification technology has many applications in all systems that make use of pre-recorded speech, such as voice mailboxes or elaborate text-to-speech synthesizers. In these cases, voice conversion would prove to be a simple and efficient way to create the desired variety of speakers without the need to record different speakers. On a more medically related note, persons who suffer from some form of voice pathology or who have had some form of surgery that renders them speech impaired would find assistance in voice modification, which might restore their previous speaking capabilities. In the same sense, with inter-national communication becoming more and more commonplace, work is being done to recognize (ASR) and translate utterances from one language to another (also known as Machine Translation) and re-synthesizing the translated utterance. Voice conversion would be of valuable assistance in preserving the naturalness of the re-synthesized speech.

All in all, we can see that our problem of voice conversion is directly related to the advancement of *human language tecnologies*, including the aforementioned automatic speech recognition, machine translation, and natural language processing, as well as speaker identification, and telephony in general. Perhaps the most significant distinction between our topic and the rest of those at hand is that, in voice conversion, we are ultimately interested in the quality of the re-synthesis of speech, targeted towards a human listener [1]. This is an important distinction to keep in mind as we look ahead into the evaluation of our system.

2 System Overview

A speaker's characteristics, including speaking rate, average pitch, average pause duration between words, phrases, and sentences, and voice timbre can be summarized at fixed intervals by extracting *features*, a representation of the original signal using reduced dimensionality. We would like to be able to learn some sort of a mapping function from the features of the source speaker to the features of the target speaker.

We begin our implementation by focusing on the *text-dependent* data scenario, in which our training data is a set of pairs of speech signals $\mathcal{T} = \{(v_i^1, v_i^1)_{i=1}^m\}$ where each pair corresponds to the speakers saying the same words. In particular, v_i^s is a sequence of real numbers corresponding to the speech of speaker s on utterance i. During training, we build a Gaussian Mixture Model (GMM) that, using a variant of the Expectation-Maximization Algorithm, develops a linear mapping from the features of the source to the features of the the target. During training, the final goal for our system is: given a new speech signal v^a for speaker 1 (source), convert it to some v^b, which is supposed to correspond to speaker 2 (target) saying the same words [2].

The rest of this paper will outline the theories around which we built our voice conversion system. First, we will explain the ideas surrounding the front-end signal processing and dynamic alignment that must be done to generate a workable set of features before any training or testing. Then we will briefly discuss the graphical models that were applied to the data to build our mapping function. Next, we will go over the process of resynthesizing the converted speech and methods for overcoming the difficulties of this problem. Finally we take a look at the results generated from a few experiments and conclude with a discussion on the potential for future work.

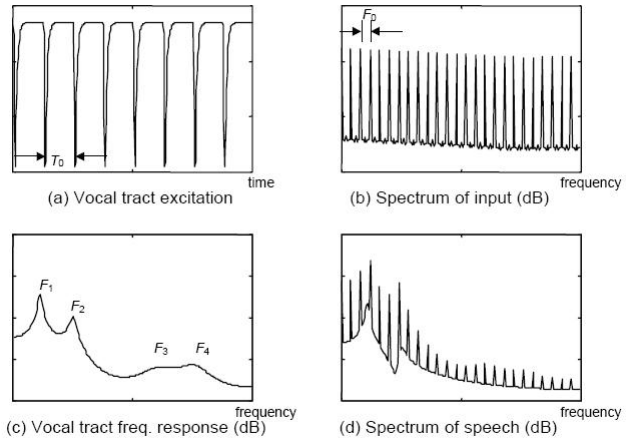

Figure 1: Plots (a) and (b) are signal and spectral plots of the typical glottal source excitation signal. Plot (c) shows the frequency response of some vocal tract configuration with resonant (formant) frequencies F_1, F_2, F_3, F_4. Plot (d) shows the resulting spectrum of the speech; in particular, $(d) = (b) \times (c)$. [5]

3 Data

The desire to work in the text-dependent scenario requires the appropriate corpus from which to run our experiments. We have been provided with the University of Wisconsin's Microbean X-ray Speech Production Database by Professor Keith Johnson of the UC Berkeley Linguistics Department. The Database is composed of more than 60 speakers each saying a specific set of utterances, ranging from word sequences to sentences and paragraphs [3]. For each speaker we have approximately 10 minutes worth of utterances, from which we devoted 6 minutes to a training set and 4 minutes to a test set.

4 Feature Extraction and Alignment

Speech can be explained by the Source-Filter Model [4]. The lungs and glottis (source) supply the power and pitch, while the shape of the vocal tract, which consists of the pharynx and the mouth and nose cavities, works like a musical instrument to produce sound. The speech we hear is a result of the resonant frequencies caused by differing shapes in our vocal tract (filter). This filter can, like all filters, be characterized by its frequency response as shown in part (c) of Figure 1. Note the resonant spectral peaks, or formant frequencies of the vocal tract.

4.1 Linear Predictive Coefficients

Based on the Source-Filter Model, one of the most fundamental feature extraction methods in speech processing are *linear predictive coefficients (LPCs)*. In this setting, we assume that the present sample of speech $x(n)$ is predicted by the past M samples of the speech such that

$$\tilde{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \cdots + a_M x(n-M) = \sum_{i=1}^{M} a_i x(n-i), \tag{1}$$

where $\tilde{x}(n)$ is the prediction of $x(n)$, and $x(n-i)$ is the i-th step previous sample. Then $\{a_i\}$ are the LPCs of the signal. As such, the error between the actual sample and the predicted one can be expressed as

$$\epsilon(n) = x(n) - \tilde{x}(n) = x(n) - \sum_{i=1}^{M} a_i x(n-i). \tag{2}$$

The objective is then to minimize the sum of the squared error with respect to each a_i.

$$E = \sum_n \epsilon^2(n) = \sum_n \left(x(n) - \sum_{i=1}^{M} a_i x(n-i) \right)^2, \tag{3}$$

This can be done by taking the corresponding derivatives of a_i and solving the resulting system of linear equations [5]; moreover, the Levinson-Durbin Recursion Algorithm can also efficiently provide a soluion to our LPC calculation [6].

With a little more work, we can also see that the Linear Predictive formulation does indeed define an all-pole filter. By slightly modifying (2) and taking the z-transform, we have

$$\epsilon(z) = X(z) - \sum_{k=1}^{M} a_k z^{-k} X(z) \tag{4}$$

Supposing now that the error (or *residual*) is the input to our system, and x is our output, we have a transfer function of the following form,

$$\frac{X(z)}{\epsilon(z)} = H(z) = \frac{1}{1 - \sum_{k=1}^{M} a_k z^{-k}} \tag{5}$$

which indeed defines an all-pole filter [7].

The key element to take away from all this is that linear predictive coefficients are a form of *feature extraction*, a representation of the original signal using fewer dimensions. Indeed, they were originally and are still widely used in compression in speech coding over telephone channels. This allows for easy quantization and transmission of the residual, which would then allow for near perfect reconstruction of the signal. Because the success of this project relies rather heavily on the quality of speech re-synthesis, LPCs were an ideal starting point.

For initial implementation, we extract a set of LPCs from a 25ms Hamming-windowed segment of the signal. To minimize cut-off errors, we do this extraction every 10ms.

A few final remarks are necessary before we conclude this discussion on linear predictive coefficients. Because they define the filter that represents a spectrum with peaks at the resonant frequencies of the vocal tract, the exact number of LPCs that are best for a given signal actually depend on the bandwidth of the speech signal. The larger the bandwidth, the more spectral peaks, and thus the more LPCs required to sufficiently model the vocal tract. In our corpus, we have speech sampled at approximately 22kHz, hence we will use an order of $22+2 = 24$ coefficients. Lastly, we should note that resonant frequencies only exist during *voiced* sounds - where the glottis actually vibrates - such as vowels. Formants do not necessarily exist for consonants, and thus we should note that the LPCs extracted for unvoiced sounds do not exactly correspond to anything with respect to the vocal tract. Nevertheless, linear predictive coefficients are effective in serving as features for the analysis and synthesis of a speech signal.

4.2 Line Spectral Frequencies

Some preliminary experimentation with creating our model to map source speaker features to target speaker features, however, showed that linear predictive coefficients, as defined above, has its limitations. Perhaps it was that the LPCs were noisy and rendered the system unable to learn a sucessful mapping function. Luckily, a look into the literature introduced the notion of *Line Spectral Frequencies (LSFs)*, which, in the speech coding domain, are often used to represent LPCs for transmission over a channel.

To arrive at LSFs, we first decompose the Linear Predictive polynomial $A(z) = 1 - \sum_{k=1}^{M} a_k z^{-k}$ into

$$P(z) = A(z) + z^{-(M+1)}A(z^{-1}) \tag{6}$$

$$Q(z) = A(z) - z^{-(M+1)}A(z^{-1}) \tag{7}$$

All the roots of P and Q lie on the unit circle and correspond directly to pole frequencies. Interestingly enough, for voiced sounds, $P(z)$ corresponds to the vocal tract with the glottis closed, while $Q(z)$ corresponds to an open glottis [8]. And though there are $M + 1$ roots for each polynomial, because of the *palindromic* symmetry of P and the *antipalindromic* symmetry of Q, the roots come in conjugate pairs $\pm w$, so $dim(LPC) = dim(LSF)$ [9]. For this final reason, LSFs are sometimes also referred to as *Line Spectral Pairs*.

In our context, LSFs perform better than LPCs for a variety of reasons. LPCs are not a very robust to noise - a small quantization error may lead to large spectral distortion - nor do they interpolate well. Because LSFs correspond directly to frequencies, they have more physical meaning and, as such, are better suited for our purposes. The implementation of this calculation is done easily in MATLAB via the command `lsf = poly2lsf(lpc)`.

4.3 Dynamic Time Alignment

In order to map features from one speaker to another, we need to create a monotonic alignment between our set of features. That is, given a training pair (v^1, v^2) that has been divided into respective feature frame vectors such that

$$v^i = (x_1^i, \ldots, x_{n_i}^i), x \in \mathbb{R}^d, i = 1, 2 \tag{8}$$

where d is the number of LSF coefficients extracted per frame, we want to find a monotonic alignment $a = \{(u^1, u^2)\}$ that minimizes the cost $\sum_{(u^1,u^2)\in a}(x_{u^1} - x_{u^2})^2$. Because the rate at which a given speaker speaks may change given differing phonetic classes of hte utterance, a linear warping of the time axis would not be ideal. Instead, we implemented the *Dynamic Time Warp* algorithm that, as its name suggests, employs dynamic programming to solve our problem [7]. Now, given the alignment, we merge adjacent frames from one signal that are aligned to one frame in the other signal. This merging is done by recalculating LSF features over a wider window, ultimately leaving us with two sequences of modified feature frames $\tilde{v}^i = (\tilde{x}_1^i, \ldots, \tilde{x}_{n_i}^i)$. Doing this for all training pairs in $\mathcal{T}$, we now have a set of frame pairs $\mathcal{F} = \{(\tilde{x}^1, \tilde{x}^2)\}$ [2].

5 The GMM-Linear Mapping of Features

Now that we have extracted the desired features, we can look into the modeling of the data and the mapping of features from source speaker to target speaker. We assume that the features of our source speaker lie in some d-dimensional space, where d is the number of LSFs extracted from a given frame, and that these features are clustered in some way related to the phonemes from which they originate. Previous work has employed vector-quantization techniques for these features, in which a codebook of mappings is generated from the training pairs and testing is done through a VQ lookup [10].

Our approach uses a softened view from the vector-quantization method; our speakers' feature space is represented in a *Gaussian Mixture Model (GMM)*. And from there, we proceed to learn a *mapping*, or conversion function.

5.1 The Gaussian Mixture Model and EM Algorithm

Given enough mixtures, a Gaussian Mixture Model should be able to approximate any arbitrary distribution. In fact, its usefulness and applicability has already been demonstrated in many applications related to speech,

from text-independent speaker recognition to speaker-independent speech recognition [7]. The GMM assumes that the probability distribution of observed parameters takes the following parametric form [1]:

$$p(x) = \sum_{i=1}^{k} \pi \mathcal{N}(x; \mu_i, \Sigma_i) \tag{9}$$

where $x \in \mathcal{R}^d$, π is a multinomial distribution over the k mixture components, and

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right\} \tag{10}$$

As such, we can say that given a source of feature vectors $\{x_t\}$, each of our features lies in some mixture of k acoustic classes $\mathscr{C}_l, l = 1, \ldots, k$ that are to be determined. Then, by an application of Bayes' rule, we can see that

$$p(\mathscr{C}_l | x) = \frac{\pi_l \mathcal{N}(x; \mu_l, \Sigma_l)}{\sum_{i=1}^{k} \pi_j \mathcal{N}(x; \mu_j, \Sigma_j)} \tag{11}$$

Now, if we iterate through the entire training set $\{x_t\}$, we will find the expected counts $c_l^{(n)}$ for mixture l:

$$c_l^{(n)} = \sum_{x \in \{x_t\}} p(\mathscr{C}_l | x) \tag{12}$$

This gives us the E-step of the Expectation-Maximization (EM) Algorithm during iteration n; we can summarize the M-step of the algorithm as follows:

Update mixture weights with relative frequency of expected counts:

$$\pi_l^{(n+1)} = \frac{c_l^{(n)}}{\sum_{i=1}^{k} c_i^{(n)}} \tag{13}$$

Gaussian means are updated below (covariances omitted):

$$\mu_l^{(n+1)} = \frac{1}{c_l^{(n)}} \sum_{x \in \{x_t\}} \tau_l^{(n)}(x) \cdot x \tag{14}$$

where $\tau_l^{(n)}(x) = p^{(n)}(\mathscr{C}_l | x)$.

The EM algorithm iteratively increases the likelihood of model parameters by successive maximizations of an intermediate quanitity which, in this case, are the conditional probabilities $p(\mathscr{C}_l | x)$ [11]. The initialization of the algorithm was uniformly random over the span of our feature space. We ran the algorithm a fixed number of iterations and used a minimal variance criterion.

5.2 Learning a mapping function

With our acoustic classes $\mathscr{C}_l$ now defined by a Gaussian Mixture Model, we can begin to learn a linear mapping for our source and target feature pairs. What comes to mind is something of the form

$$y_i = \sum_{l=1}^{k} p(\mathscr{C}_l | x_i) \times [A_l x_i + b_l] + w_i \tag{15}$$

in which we need to estimate the $(d \times d)$ linear transform A_l and $(d \times 1)$ bias vector b_l over all observed training pairs (x_i, y_i) in order to minimize the noise w_i. More specifically, we incorporate the parameters of the GMM that were just estimated [1], to give us

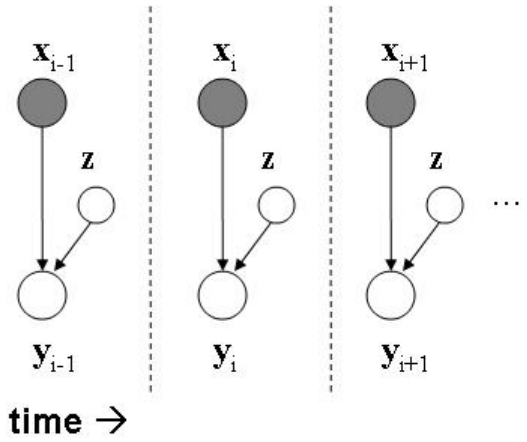

Figure 2: The frame-independent approach treats each data vector pair (x_i, y_i) as a separate model.

$$y_i = \sum_{l=1}^{k} p(\mathscr{C}_l|x_i) \times \left[A_l \cdot \Sigma_l^{-1}(x_i - \mu_l) + b_l\right] + w_i \tag{16}$$

From now on, to simplify notation, let us denote $\bar{x}_i^{(l)} = \Sigma_l^{-1}(x_i - \mu_l)$. Thus, the problem of minimizing the error resembles that of a least squares linear regression. By solving this directly, we can obtain our voice conversion function.

5.3 Increasing Model Complexity

Evaluation of this initial system produced decent results. The results from a subjective listening test using 3 evaluators with limited knowledge of the project suggest that, modulo the quality of synthesized speech, the converted speech resembles, at best a 50/50 blend between the source and target speakers.

However, our initial GMM and linear mapping was based on a very limited model. We had employed the naive frame-independent approach as shown in Figure 2. Such a "bag of frames" methodology resulted in the following linear mapping:

$$y_i = \sum_{l=1}^{k} p(\mathscr{C}_l|x_i) \times \left[A_l \cdot \bar{x}_i^{(l)} + b_l\right] + w_i \tag{17}$$

It made sense, however to also try incorporating the neighboring data pairs into the Model, resulting in another linear mapping function and the graphical model as depicted in Figure 3:

$$y_i = \sum_{l=1}^{k} p(\mathscr{C}_l|x_{i-1}, x_i, x_{i+1}) \times \left[A_l\bar{x}_i^{(l)} + b_l + C_l\bar{x}_{i+1}^{(l)} + D_l\bar{x}_{i-1}^{(l)}\right] + w_i \tag{18}$$

This approach worked better, improving slightly upon subjective evaluation. What seemed to be missing, however, was the notion of time dependence in the output. The final enhancement we made, then, was to add a sort of 'Markovian' dependence to the output sequence such that our estimated target features $y(n)$ are dependent on the previously estimated target features y_{n-1} as well as the present and future source features $x(n), x(n+1)$. The corresponding model is depicted in Figure 4, and the resulting linear mapping function is of the form:

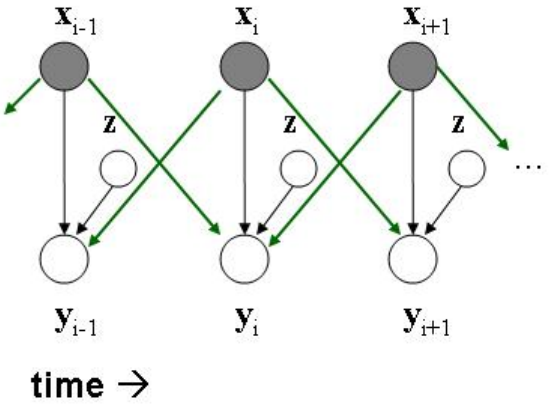

Figure 3: This approach uses data locally observed to map $\mathbb{R}^{3d} \rightarrow \mathbb{R}^d$. This is the locally dependent approach.

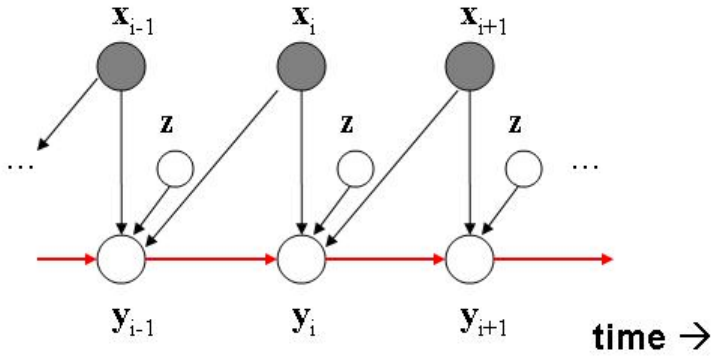

Figure 4: This approach uses data locally observed as well as previously estimated data to map $\mathbb{R}^{3d} \rightarrow \mathbb{R}^d$. We call this the Markov dependence approach.

$$y_i = \sum_{l=1}^{k} p(\mathscr{C}_l | y_{i-1}, x_i, x_{i+1}) \times \left[A_l \bar{x}_i^{(l)} + b_l + C_l \bar{x}_{i+1}^{(l)} + D_l \bar{y}_{i-1}^{(l)} \right] + w_i \tag{19}$$

Unless otherwise specified, we used this model for most of the experiments discussed in this paper. Sometimes, due to unforeseen errors or computational complexity issues, it was faster to get things going with the initial frame-independent approach at little cost to our results. In any case, the final implementation of this graphical model involved training a mixture of $k = 16$ Gaussians with 10 iterations of the Expectation-Maximization algorithm.

6 Synthesizing Converted Speech

In our discussion of Linear Predictive Coefficients and the Source-Filter Model, it was seen that if the appropriate residual error values ϵ are applied to the filter defined by LPCs $\{a_i\}$, we can obtain a near-perfect reconstruction of of the original signal. This works, of course, when the residual can indeed excite the resonant frequencies of the filter. In voice conversion, given a set of predicted features, discovering the proper residual for high quality, target-speaker-sounding speech is a difficult problem in itself. Here, this process is known as *residual prediction* [6].

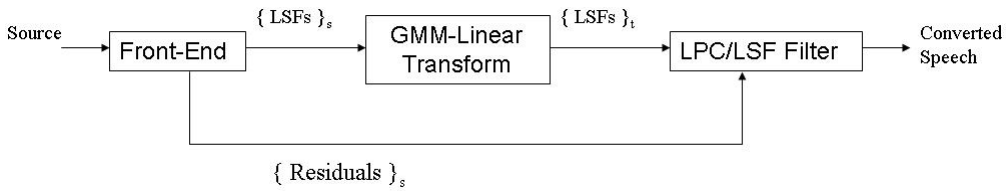

Figure 5: Schematic diagram for copying source residuals to synthesize converted speech.

6.1 Copying Source Residuals

A number of interesting methods have been proposed. The simplest one, in fact, does no actual prediction at all; it is to simply *copy source residuals*. Technically speaking, the ideal source-filter model assumes that a large majority of speaker-dependent information can be represented by the vocal tract and, hence, the extracted features. We expect the source excitation to be less crucial; hence it makes sense to just use the source residuals and apply the transformed features [6].

As depicted in Figure 5, this was the first method implemented into the initial voice conversion system. Its result can be witnessed on Tracks 1-3 (Source, Target, Converted) in the accompanying demo CD. The quality of the utterance, while a bit crude and unpolished, was actually somewhat better than initially expected. The converted speech was, for the most part, intelligible and had at least a slight resemblance to both speakers. Nevertheless, it is clear that merely changing the parameters of our filter is not enough to change a speaker's identity. At the end of it all, we might say that a third speaker was created in this process [12].

The shortcomings of this method were made more pronounced in a more ambitious voice conversion task: cross-gender. Tracks 1-3 demonstrated a conversion between two female speakers. Tracks 4-6 (Source, Target, Converted) show the system's lack of robustness in converting from a male voice to a female voice. This prompted an investigation of other approaches to residual prediction.

6.2 Residual Codebook and Selection

In an effort to avoid using the source speaker's parameters in the converted utterance, the next logical approach would be to build a codebook that stores the target's feature vectors y_m seen in training with the corresponding residuals r_m that were seen in training. Then during synthesis, given a predicted feature vector $\tilde{y}$, we can select the residual from the codebook whose corresponding feature vector minimizes the squared error between $\tilde{y}$ and all feature vectors $\{y_m\}$ seen in training [13]. That is, choose

$$\tilde{r} = r_{\tilde{m}} \text{ where } \tilde{m} = \arg\min_m .|\tilde{y} - y_m| \tag{20}$$

While it sounds promising, we ought to keep in mind that selecting from disjoint and discontinuous residuals often causes problems in phase mismatch, among others, which would introduce unwanted disturbances into the synthesized utterance. Unfortunately, due some shortcomings in computing capability, we were unable to test out this approach in time for the completion of this paper. It would, nevertheless, be interesting to see how the results of this method compare with results of the previous.

6.3 The Vocoder Approach

The vocoder approach is based on the standard formulation of lineear predictive coding, where the source residual signal is either white noise or a pulse train that resembles unvoiced or voiced excitations, respectively [6]. This approach, as depicted in Figure 6, may seem like a step backwards from the approaches mentioned above; indeed, vocoders have a reputation for creating less natural, more synthetic sounding speech. However, there have been recent advances to the vocoder in the STRAIGHT project, which managed to overcome the monotony of a synthetic sounding voice via techniques involving natural residual waveforms.

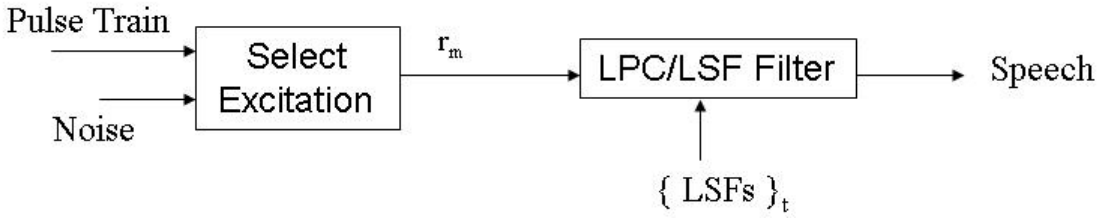

Figure 6: Schematic diagram for the synthesis of converted speech using the vocoder approach.

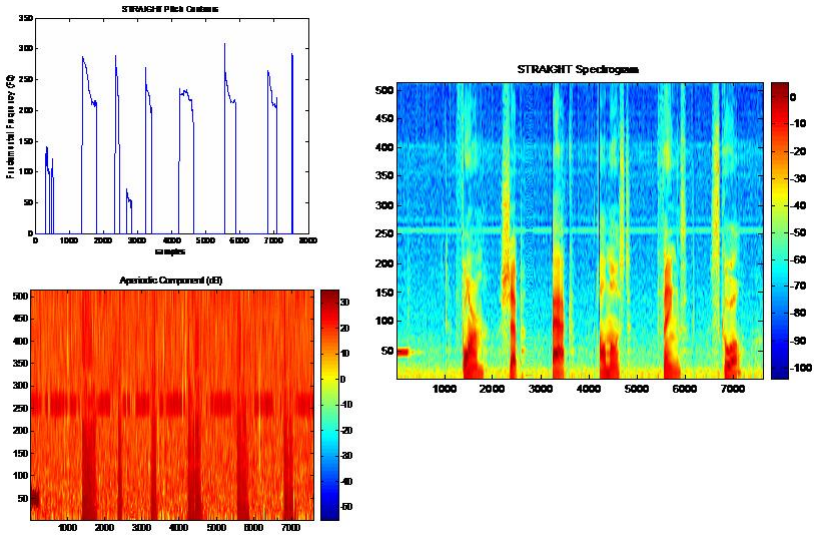

Figure 7: Parameters that are extracted by the STRAIGHT system: (Top-Left) A vector of F0 values, (Bottom-Left) The Aperiodic component of the speech, (Right) A smoothed Spectrogram.

7 The STRAIGHT Vocoder

The STRAIGHT (Speech Transformation and Representation by Adaptive Interpolation of weiGHTed spectrom) system was originally designed and built by Professor Hideki Kawahara to investigate human speech perception. It was also motivated by the need for flexible speech modification, the simplicity of the channel vocoder methods (source-filter separation), and the lack of natural quality in the resulting speech produced by such a method [14]. The result of this project was a system whose reproduced speech sounds after modification are sometimes indistinguishable from the original speech sounds in terms of naturalness [15].

STRAIGHT uses procedures that can be grouped into three subsystems: a source information extractor, a smoothed time-frequency representation extractor, and a synthesis engine consisting of an excitation source and a time varying filter [16]. The elements that are extracted from the first two subsystems are depicted in Figure 7. And to witness the ability of the synthesis, Tracks 7-8 of the accompanying demo CD consist of an original utterance (Track 7), while Track 8 is the result of using STRAIGHT to extract information and then using that as the input to the synthesis engine. Before we dive any deeper, we should realize that the STRAIGHT system has no desire for information reduction. Because the quality and flexibility for manipulations was the main focus for its development, we need to be wary that dimensionality and computability become a concern when trying to scale STRAIGHT for use in our own voice conversion system.

The following subsections will provide a summary of the methodology used in the STRAIGHT system

to provide high quality speech analysis-modification-synthesis based on the channel vocoder formulation.

7.1 F0 Extraction

For the ability to model, adjust, and reproduce speech, it is important to be able to extract F_0 trajectories which do not have any trace of interferences caused by the length of the analysis window and the signal waveform. Pitch extraction algorithms based on the usual definition of periodicity do not behave well for this purpose, because a natural speech signal is neither purely periodic nor stable [14]. To work around this issue, the STRAIGHT system extracts as the fundamental frequency the instantaneous frequency of the fundamental component of the signal.

The F_0 estimation method of STRAIGHT assumes that the signal has a nearly harmonic structure, as follows,

$$x(t) = \sum_{k=1}^{N} a_k(t) \cos\left(\int_0^t (k w_0(\tau) + w_k(\tau)) d\tau + \phi_k(0) \right), \tag{21}$$

where $a_k(t)$ represents a slowly changing instantaneous amplitude, and $w_k(\tau)$ also represents slowly changing perturbations of the k-th harmonic component. As such F_0 is the instantaneous frequency of the fundamental component where $k = 1$ [16]. To extract the fundamental component, we apply a series of band-pass filters arranged in log-linear fashion (6-24 per octave) and use them to extract fixed points that map from the filter center frequency to the instantaneous frequencies of the filter output. The filter impulse response $w_F(t, \lambda)$ is composed of a Gabor function $w(t, \lambda)$ convolved with a 2nd-order cardinal B-spline basis function $h(t, \lambda)$ than can be tuned to some frequency λ. Hence, we have

$$w_F(t, \lambda) = w(t, \lambda) \otimes h(t, \lambda), \tag{22}$$

$$w(t, \lambda) = e^{-\frac{\lambda^2 t^2}{4\pi\eta^2}} e^{j\lambda t}, \tag{23}$$

$$h(t, \lambda) = \max\left\{ 0, 1 - \left| \frac{\lambda t}{2\pi\eta} \right| \right\} \tag{24}$$

which is essentially a continuous wavelet transform [17].

Now, if we can tune λ such that $\lambda = 2\pi F_0$, then this filter will effectively suppress interference from neighboring harmonic components. To do so, we seek a set of fixed points $\{\lambda_s^*\}$ such that a filter $w_s^*(t, \lambda^*)$ with center frequency λ_s^* will have an output with instantaneous frequency $w_c(t, \lambda_s^*) = \lambda_s^*$ [17]. The F_0 that is selected is the one having the distinctly higher signal to noise ratio of sinusoidal component and background noise [16]. In the event of no distinct fundamental frequency (e.g. pauses in between words, unvoiced sounds), the expected pitch value of 0 is returned.

The above outlined the procedure for an initial estimate of F_0, which obtains reasonable accuracy. However, this process can be improved by a refinement procedure that uses the initial estimates to perform another iteration that finds fixed points corresponding to harmonic components that can, once again, use a signal to noise ratio to provide an updated estimate with minimum estimation error.

A look at the MATLAB implementation of STRAIGHT provides some additional insight into this methodology. As part of its default parameter settings for extracting F_0 information, the STRAIGHT system searches over a default window length of 40ms for possible F_0's of between 40 and 800Hz. Performance is said to improve if this search range can be decreased, that is, if we have a priori knowledge of the speaker's F_0 range [18].

7.2 Aperiodicity Extraction

Previously, we saw that the F_0 estimation method of STRAIGHT assumes a nearly harmonic structure of the signal. However, there will always exist deviations from periodicity, which introduce additional components

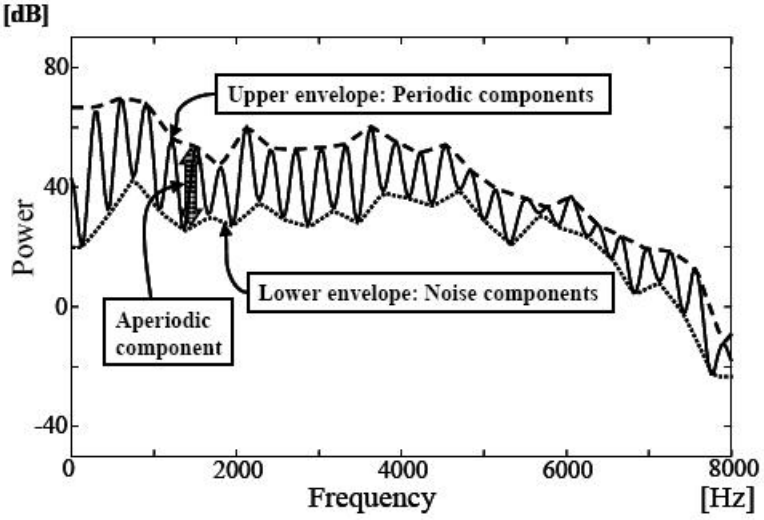

Figure 8: Aperiodic component extraction [19].

on inharmonic frequencies. As such, we can find a measure of aperiodicity by taking the energy on inharmonic frequency normalized by the total energy.

During the extraction of the fundamental frequency , STRAIGHT creates a window function by convolving a slightly time-stretched (η) Gaussian with a 2nd-order cardinal B-spline function (24) that is tuned to the fixed F_0 [16]. This window is designed to have zeros between harmonic components; thus a power spectrum calculated using this window provides the energy sum of periodic and aperiodic components at each harmonic frequency and provides the energy of the aperiodic component at each in-between harmonic frequency.

To summarize the procedure, let $|S(w)|^2$ represent a power spectrum of an utterance, and then let $|S_U(w)|^2$ and $|S_L(w)|^2$ represent the upper and lower spectral envelopes respectively. The upper envelope is calculated by connecting spectral peaks, while the lower envelope is calculated by connecting spectral valleys, as depicted in Figure 8. The *aperiodicity measure* of a certain center frequency w (as integrated over some window of neighboring frequencies $W(w)$ is then defined as the lower envelope normalized by the upper envelope as such

$$P_{AP}(w) = \frac{\int W(w)|S(\lambda)|^2 \left(\frac{|S_L(\lambda)|^2}{|S_U(\lambda)|^2}\right) d\lambda}{\int W(w)|S(\lambda)|^2 d\lambda} \tag{25}$$

This aperiodic measure provides additional information regarding the speaker's source that is not summarized in the F_0 extraction. While it is mostly true that most of a speaker's prosodic features can be well represented in the previously discussed F_0 extraction and the to-be discussed time-frequency spectral envelope, the aperiodicity measure that was just summarized still contains information that is perceptually significant.

7.3 Spectrogram Extraction

Speech is inherently periodic and, ironically, this does pose problems. It is, in some ways, contradictory and frustrating that periodicity induces such difficulty in speech analysis and manipulation; for human listeners, voiced sounds are perceived to be smoother and richer than unvoiced sounds [14]. And yet, what we now

seek in this problem of spectrogram extraction is to obtain a time-frequency representation that does not have any trace of periodicity.

When the length of a time window for spectral analysis is comparable to the fundamental period of the signal repetition, the resultant power spectrum shows periodic variation in the time domain. Conversely, when the length of the time window spans several repetitions of this fundamental period, the resultant power spectrum shows periodic variation in the frequency domain [14]. What we seek, then is a perfectly sized rectangular window that is equal to the fundamental period such that variations in either domain are not apparent. Unfortunately, this is not possible in the context of natural speech. Fundamental frequencies of these signals change all the time; moreover, the sharp discontinuities of a rectangular window make these representations highly sensitive to minor errors.

The most unique feature of STRAIGHT is its ability to perform extended pitch synchronous analysis without requiring, as other pitch synchronous procedures do, that its analysis frame be aligned to the pitch in any specific manner. Work for this has already been done in the process of F_0 extraction. A similar filter from (22) is used, resulting from a convolution of an isotropic Gaussian with the 2nd-order cardinal B-spline basis $h(t, \lambda)$ as copied from (24):

$$w_p(t, \lambda) = e^{-\frac{\lambda^2 t^2}{4\pi\eta^2}} \otimes h(t, \lambda), \tag{26}$$

$$h(t, \lambda) = \max\left\{0, 1 - \left|\frac{\lambda t}{2\pi\eta}\right|\right\} \tag{27}$$

where again we have tuned $\lambda = 2\pi F_0$, and allowed for some time stretching η for improved frequency resolution [15]. This places the second-order zeros on the other harmonic frequencies, which makes the resultant spectrum less sensitive to any F_0 extraction errors [14].

The compensatory window w_c that produces peaks at positions where zeros were located in (22) (i.e. fills in the "holes") is achieved by the following modulation:

$$w_c(t) = w_p(t) \sin\left(\frac{\lambda t}{2}\right) \tag{28}$$

Then finally, the resulting composite spectrum $P_r(w, t, \eta)$ can be represented as a weighted squared sum of the power spectra $P_o^2(w, t, \eta)$, using the original time window, and $P_c^2(w, t, \eta)$, using the compensatory window.

$$P_r(w, t, \eta) = \sqrt{P_o^2(w, t, \eta) + \xi(\eta) P_c^2(w, t, \eta)} \tag{29}$$

where ξ is selected to minimize temporal variation of the resultant spectrogram. This result, in addition to some additional smoothing procedures, allows for pitch synchronous spectral analysis and the creation of the "STRAIGHT spectrum" [15].

7.4 Synthesis

The final element left to discuss is that of STRAIGHT synthesis. However, now that we know the source (F_0, aperiodic component) and filter (time-frequency spectral envelopes) parameters, there is not much to do but to place it all back together and get back the output speech. Figure 9 shows a schematic diagram of how STRAIGHT is used.

One of the big pitfalls in speech synthesis following the vocoder framework is the inherent "buzzy" sound that results from a pulse-like excitation [15]. To combat this effect, STRAIGHT introduced a group delay manipulation to enable user control of the F_0 that is finer than the resolution determined by the sampling interval. While the mathematical details are explained in detail in [20], we can summarize by saying that the resynthesis engine in STRAIGHT includes an all-pass filter generated via random numbers as well as the introduction of asymmetry into the group delay to produce more interesting timbre. The final signal is generated by a pitch-synchronous overlap add of the resulting source signal convolved with the minimum phase impulse response calculated from the extracted (and possibly modified) spectral envelopes.

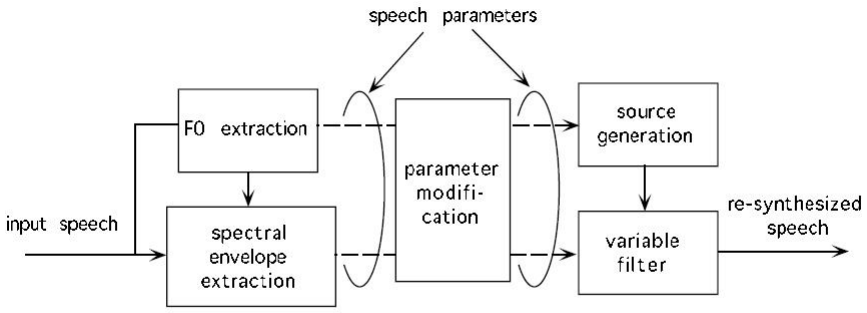

Figure 9: A schematic diagram of the STRAIGHT system. Note, howe,ver, that In our implementation, we are doing "conversion" instead of "modification" [14].

8 Integrating STRAIGHT

The intricacies and details within the STRAIGHT system took some time to understand. One of the original goals of the project this semester was to actually get into the STRAIGHT source code and extract the relevant parts to improve our Voice Conversion System. Unfortunately, for right now, we will have to settle with the ability to merely integrate relevant parts of STRAIGHT into the present system.

8.1 Implementation

Our initial approach to resynthesis merely copied source residuals which, because they do not properly excite the filter parameters at every given frame of speech, resulted in a very noisy sounding resynthesis. As such, our top priority was to use the STRAIGHT resynthesis engine, which meant we would need to be able to provide all three aforementioned components (F_0, Aperiodicty, Spectrogram) in proper STRAIGHT format to the synthesizer.

Pitch was easy to handle. In fact, we had previously neglected to directly compensate for differences in pitch between our source and target speaker, which was at least a partial reason in our failed attempt to convert a male voice into that of a female (Tracks 4-6 in Demo CD). So we began by figuring out the bias between the source pitch p_s and target pitch p_t. That is, we learned the function $p_t(n) = p_s(n) + b$ where $b = \bar{p_t} - \bar{p_s}$ is the difference between the means of the source and target pitches seen in training. We also began looking at a way to modify pitch trajectories in order to model the differences in vocal inflection between the speakers. However, the implementation could not be debugged in time to report on its results.

At this time, we decided to not worry so much about generating or learning a conversion function between aperiodic components. As such, instead of copying the source residuals per se, we copied the source aperiodic component and hoped that the fundamental frequency and spectrogram mapping would suffice in a successful conversion.

The generation of a spectrogram was an interesting problem. As mentioned previously the STRAIGHT approach makes no attempt for information reduction [14]. Unfortunately, to learn a mapping function in some finite amount of time, we could not afford to let dimensionality get out of hand. The STRAIGHT system extracted a 512-point spectral envelope every 1ms over a 40ms window, while our conversion system extracted a set of 24 LSFs every 10ms over a 25ms window. In an slight effort to compromise, we began extracting and mapping LSF vectors of 24 dimensions every 5ms over a 40ms window. To create the spectrogram, we would then use noise to excite the spectral envelope parametrized by the converted LSFs and simply smear it over 5ms (i.e. MATLAB: `repmat(SpecEnv,1,5)`).

8.2 Results, Analysis, and Further Work

Though much of the theory suggested that everything would work out, the results were less satisfying than expected. Though the quality of speech may have been perhaps slightly better than the original method of copying source residuals, the signal still sounded very noisy and unclear. Our best result was the ability to make the male-to-female conversion at the very least intelligible. The result of said conversion using the STRAIGHT implementation can be heard on Track 9 (Source: Track 4, Target: Track 5), while the result from a female-to-female conversion (Source: Track 1, Target: Track 2) can be found on Track 10.

Perhaps there is an underlying bug that has plagued our implementation from the beginning; it is unclear where the problem truly lies. If one listens hard enough a sufficient number of times, he or she could probably say the converted speech resembles parts of both speakers, but at the end of the day, we have mostly just created a third speaker speaking in a noisy, artifact-filled environment.

There are many directions in which we could progress; however, we might begin by looking more introspectively and begin with experiments that would bring us closer to explaining the shortcomings of our solution. The STRAIGHT project is an incredibly useful tool and it would be great to work with it more and understand it in greater detail. Perhaps we ought to look a little deeper into the mapping of aperiodic content and into our attempt to represent the smoothed, pitch-synchronous spectrogram using just LSFs. It may even be beneficial to the ongoing STRAIGHT progress to work on methods of information reduction for faster real-time implementation of STRAIGHT [15]. Wherever we decide to move next with this project, there will be potential for improvement.

9 Discussion

And looking past the immediate problems, there are many other areas that can be explored to improve our voice conversion results. In addition to finishing up learning pitch trajectory mapping, we could look into durational differences between speakers. Even though a dynamic time-alignment was performed to pair up the feature vectors, we have yet to do anything to warp the time axis during the actual conversion phase.

That might, in fact, lead into a more complicated system of feature extraction and graphical modeling in general. On a note related to this project, there has already been some work done to incorporate a GMM model with the STRAIGHT excitation to generate a converted residual signal based on Maximum Likelihood Estimation [19]. Other work has employed a Hidden Markov Model (HMM) to model the dynamic characteristics of a speaker, where the HMM has a state-dependent codebook of feature mappings [21]. Unfortunately, the codebook approach has proven to be outdated, but recently, there was work done in combining an HMM with the GMM model. Analogous to the graphical model framework in automatic speech recognition, this HMM does not use phonemes as states; rather, it uses utterance and speaker specific information trained from the GMM to determine its states and transition probabilities [22]. Following the work in this area is a definite possibility for the future.

10 Conclusion

Our ultimate goal is to be able to do voice conversion in the text-independent setting, the form of unsupervised learning where training utterances from the target and source speakers need not be aligned. Until we get to that point, however, we will be able to keep ourselves busy working on applying audio signal processing and machine learning techniques to the text-dependent problem. This project turned out to be an enriching opportunity to further investigate the complexities of speech outside of the ASR domain. I had a lot of fun playing with sound files and see potential for continuing work on this problem in the future.

11 Acknowledgements

For the training and testing of our system, we were generously provided with the University of Wisconsin's Microbeam X-ray Speech Production Database by Professor Keith Johnson of the Linguistics Department at UC Berkeley. Our data manages to provide a lot more than just same-text speech for over 60 speakers; the Microbeam X-ray technology used provides data for movement of various articulators in speech, which is incredibly useful for work in all areas of linguistics and speech processing. We do apologize for not using this corpus at its fullest potential, though we are, nonetheless, grateful to have been granted access to this data.

And for guiding an undergraduate into this project, I would like to thank EECS Graduate Student Percy Liang for his patience in letting me badger him about the machine learning code (for the GMM-Linear Mapping) he provided. I appreciate his assistance in that regard, as I had my hands full trying to figure out how to put everything else in place. Also a thanks to EECS Graduate Student Suman Ravuri for introducing to us the STRAIGHT system; I cannot imagine what it would have been like trying to figure it all out from scratch.

Finally, a thank you to Professor Nelson Morgan for supporting this work-in-progress as my EE 225D course project.

References

[1] Yannis Stylianou, Olivier Cappe, and Eric Moulines. Continuous probabilistic transform for voice conversion. *IEEE Trans. on Speech and Audio Processing*, 6(2):131–142, 1998.

[2] Percy Liang and Stephen Shum. Voice conversion notes - version 1. *Updated November 6, 2008.*

[3] John R. Westbury. *X-ray Microbeam Speech Production Database User's Handbook.* University of Wisconsin, 1994.

[4] Keith Johnson. *Acoustic and Auditory Phonetics.* Blackwell, Malden, Massachusetts, 2003.

[5] S. Park. Dsp, chapter 7. *www.engineer.tamuk.edu/SPark/chap7.pdf.*

[6] David Suendermann. *Text Independent Voice Conversion.* PhD thesis, Bundeswehr University Munich, 2007.

[7] Nelson Morgan and Ben Gold. *Speech and Audio Signal Processing.* John Wiley and Sons, Inc., New York, 2000.

[8] Tony Robinson. Line spectral pairs, 1998. *http://svr-www.eng.cam.ac.uk/ ajr/SpeechAnalysis/node51.html.*

[9] Jonathan Y. Stein. *Digital Signal Processing: A Computer Science Perspective.* Wiley-Interscience, New York, 2000.

[10] Masanobu Abe, Satoshi Nakamura, Kiyohiro Shikano, and Hisao Kuwabara. Voice conversion through vector quantization. In *Proc. IEEE Int Conf. Acoustics, Speech, Signal Processing*, 1988.

[11] Michael I. Jordan. Introduction to probabilistic graphical models. Berkeley, CA, 2003. Unpublished.

[12] Alexander Kain and Michael W. Macon. Spectral voice conversion for text-to-speech synthesis. In *Proc. IEEE Int Conf. Acoustics, Speech, Signal Processing*, 1998.

[13] Hui Ye and Steve Young. High quality voice morphing. In *Proc. IEEE Int Conf. Acoustics, Speech, Signal Processing*, 2004.

[14] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain de Cheveigne. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction. *Speech Communication*, 27:187–207, 1999.

[15] Hideki Banno, Hiroaki Hata, Masanori Morise, Toru Takahashi, Toshio Irino, and Hideki Kawahara. Implementation of realtime straight speech manipulation system: Report on its first implementation. *Acoustical Science and Technology*, 28(3), 2007.

[16] Hideki Kawahara, Jo Estill, and Osamu Fujimura. Aperiodicity extraction and control using mixed mode excitation and group delay manipulation for a high quality speech analysis, modification and synthesis system straight. In *Proc. of the MAVEBA, Firenza, Italy*, 2001.

[17] Hideki Kawahara, Haruhiro Katayose, Alain de Cheveigne, and Roy D. Patterson. Fixed point analysis of frequency to instantaneous frequency mapping for accurate estimation of f0 and periodicity. In *Proc. Eurospeech*, 1999.

[18] Hideki Kawahara. Tips to make re-synthesis quality better. STRAIGHT Trial Webpage. *http://www.wakayama-u.ac.jp/ kawahara/STRAIGHTtrial/betterresynth.html*.

[19] Yamato Ohtani, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Maximum likelihood voice conversion based on gmm with straight mixed excitation. In *Proc. InterSpeech*, 2006.

[20] Hideki Kawahara. Speech representation and transformation using adaptive interpolation of weighted spectrum: Vocoder revisited. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[21] E. Kim, S. Lee, and Y. Oh. Hidden markov model based voice conversion using dynamic characteristics of speaker. In *Proc. European Conference On Speech Communication and Technology*, 1997.

[22] Yue Z., Zou X., Jia Y., and Wang H. Voice conversion using hmm combined with gmm. In *Proc. Congress on Image and Signal Processing*, 2008.