Speech Enhancement Using Synthesis and Adaptive Techniques

James Fosburgh
Audio and Music Engineering Department
University of Rochester
Rochester NY, USA
jfosburg@u.rochester.edu

Scott Bradley
Audio and Music Engineering Department
University of Rochester
Rochester NY, USA
sbradl10@u.rochester.edu

***Abstract*—In this paper, we propose a method for speech signal enhancement by estimating the clean speech signal through the use of a speech recognition system. The idea is based on the observation that noisy/reverberant speech is easier to comprehend if the listener knows the language and knows what likely combinations of words are. Therefore, we surmise that if a computer has an idea of what the clean output of the noisy speech will likely sound like, it will be able to do a better job of removing the noise. The overall approach to our method is as follows: detect the phoneme and pitch of the corrupted speech signal for each frame, use this information to synthesize a rough approximation of the clean speech signal, then use this signal to estimate the noise from the corrupted signal and use spectral subtraction to get the final clean output.**

***Index Terms*—spectral subtraction, unit selection synthesis, automatic speech recognition**

I. Introduction

Speech enhancement is used in a wide variety of applications. The most notable applications include telecommunications, speech recognition, music/sound production. With the rise of importance of voice assistants that use speech recognition, it is important for the systems to have clean speech data in order to improve functionality. Our system of speech enhancement is a general purpose system. It can be applied to any desired field, and is meant to improve a wide range of speech in different contexts.

A. Existing De-Reverberation and De-Noising Methods

Spectral Subtraction is a technique that is used in many different contexts. Spectral subtraction can be described as the process of restoring a signals magnitude spectrum through the subtraction of an estimated noise component [1]. The noise is estimated in several different ways, and depends on the context of the implementation. Different noisy signals will expect different noise, and the noise estimation and subtraction will need to be updated given the tasks constraints. An early example of spectral subtraction was used for speech dereverberation [2]. In this algorithm, they enhance a noisy speech signal that has been corrupted with reverberations, or echoes. they use the principles of room acoustics to estimate the power spectral density (PSD) of echoed voice, and look at past frames of data to see which phonemes would be echoed onto the current signal. Once they estimate this noise they can subtract it from the original signal to result in dereverberated speech. This implementation of spectral subtraction is very specific, and is only applied to a very narrow set of contexts. This application is not meant to remove any noise other than reverberant noise, and will only perform well when that is the only noisy component of the signal.

As neural networks have grown in popularity over the last decade, the trend of using them to solve signal processing problems has not skipped over speech enhancement. In 2015, Han *et al.* [6] proposed a method of using deep neural networks (DNNs) to learn the mapping of reverberant and noisy speech to clean speech. In their approach, they used the short-time Fourier-transform of the current frame of the input signal and the five frames on either side of it as the input to their neural network.

In 2016, Xiao *et al.* [7] also presented a similar approach using DNNs that also incorporated more control over the process. Xiao *et al.* used a restrictive Boltzmann machine to evaluate the noisy speech signal and initialized the DNN coefficients using the information learned. They also included the Delta (first-order time derivative) and Acceleration (second-order time derivative) of the noisy signal in the cost function of the DNN in an attempt to improve the continuity of the output. Both papers noted that their methods resulted in significant improvement of signal quality, but cited the need to more fully test the performance on signal conditions that deviate more and more from the training conditions.

B. Speech Synthesis and Recognition Techniques

Many approaches exist to solving speech synthesis. The newest and most complicated method is using neural networks. In this approach, a neural network is trained on recorded speech data from a user and essentially learns how that persons voice works. Other methods include: concatenation synthesis, in which individual recorded sounds are strung together; formant synthesis, in which a physical model is used in conjunction with additive synthesis to create a waveform of speech; articulatory synthesis, which is based on physical models of the human vocal tract and the way it produces articulations; and HMM-based synthesis, in which the frequency spectrum of speech sounds, pitch, and duration

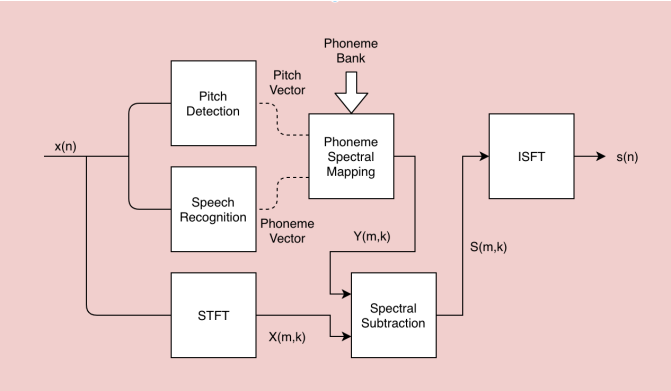

Fig. 1. Block diagram showing the flow of our proposed approach.

are all modeled by Hidden Markov Models. In this method we will be implementing concatenative synthesis with the unit being individual phonemes. We chose this method for ease of implementation, both in terms of complexity and training time/data.

One of the most widely used general-purpose approach to automatic speech recognition (ASR) is an HMM based approach. HMMs work well for ASR as speech signals can be broken down into piece-wise chunks of information. The generally accepted approach to HMM-based ASR is to break down the speech signal into individual frames and compute the cepstral coefficients of the signal at that frame. Then, the HMM uses this coefficient vector to assign likelihood values to the denotation units used in the model (phonemes, diphones, words, etc.). As the focus of our project is not on ASR, we will be using an existing ASR toolkit - *CMUSphinx* - which is an open-source HMM-based ASR toolkit developed at Carnegie Mellon University.

II. Proposed Approach

A. Automatic Speech Recognition

For our implementation, we need ASR on the level of individual phonemes. While this may initially seem easier than detection of whole words, it is in fact less accurate. The reason for this is that when you look on the level of whole words, you can use context clues to improve the accuracy. When only looking at the phoneme level, there is more ambiguity on what a certain sound will be identified as, as many phonemes have similar sounds. While this may seem like a problem, we do not believe this will result in a significant decrease in quality with the end result being de-noising. This is because, while it will decrease the accuracy of speech synthesis as a whole, it is likely that misidentified phonemes will have similar spectral makeups to the actual phonemes. Because our de-noising algorithm is based on the similarity in spectral makeup between the speech from the noisy signal and that of our synthesized speech, the final result should not be significantly affected.

Through CMUSphinx, we have access to a state of the art ASR system. More specifically, we will be using PocketSphinx: a version of the Sphinx program that has been developed for use in embedded systems. The output of this program is the list of phonemes present in the signal in order of occurrence, as well as the times at which they start. This, combined with the pitch vector found from the following step in the process, will be fed into our concatenative synthesis algorithm to produce the estimated clean speech signal.

B. Pitch Detection

In our approach, we have elected to use the Yin pitch detection algorithm. The Yin algorithm is a time-domain based pitch detection algorithm commonly used for pitch detection in speech and music. It is implemented by finding the lowest points in a difference function taken by subtracting a time shifted version of the signal from itself. A more detailed explanation of the Yin algorithm and the steps for implementing it can be found in [8].

C. Speech Synthesis

With our pitch and phoneme vectors computed, we can start the estimated speech signal synthesis. In our project, we will be using a form of concatenative synthesis known as unit selection synthesis. In this process, speech is synthesized by concatenating successive chunks of pre-recorded speech sounds. In our case, we have recorded samples for each phoneme in the English language.

1) Phoneme Look-up: In our algorithm, we will be going through frame by frame of the corrupted signal. The first step is to determine which phoneme is present in the signal at that frame. This will be done by checking the current time of that frame with regards to the start of the signal and looking up in the phoneme vector which time-stamps that frame falls between. We then retrieve that phoneme from our recorded phoneme bank for modification.

2) Pitch Modulation: The next step is to modify the pitch of the phoneme based on the corresponding pitch from the pitch vector. As each phoneme in our bank is also denoted with its frequency, we know the ratio between current and desired pitch. With this knowledge, we can shift the pitch of our sample using the phase vocoder process. This involves resampling the phoneme to achieve the desired change in pitch, then stretching or shrinking the signal to make up for the duration change due to resampling.

3) Concatenation: Once we have our sequence of pitch-shifted phonemes, we concatenate the phonemes to synthesize our estimated clean signal. To do this, we simply use a weighted overlap-add technique to combine successive phonemes. We chose to use an overlap-add method of concatenation for two reasons: it will produce a smoother sounding output than just tacking phonemes on one after another, and it will also provide for smoother transitions between different pitches and phonemes.

D. Spectral Subtraction

1) Overview: In our implementation of spectral subtraction, we were able to use the other system components to create a more accurate noise estimation. Since we have a generated speech signal, we can use that as an estimate of what the final clean signal should be. Using the estimated speech signal and the original speech signal, we can find a very accurate error between the two signals that corresponds to the continuous noise signal. By combining this estimate with traditional noise estimation techniques, we can choose the best noise estimate that accurately describes all types of noise. With this final noise signal, we subtract it from the original signal and the difference is a clean speech signal.

2) Implementation: We start by taking the short-time Fourier transform of both the original signal x(n) and the synthesized speech signal y(n) to generate X(m,k) and Y(m,k), where m is the frequency bin and k is the frame. We use a hamming window of 512, and a hop size of 128 points. Using this, we estimate the initial noise PSD N:

$$N(1,k) = mean(|X(m,k)| - |Y(m,k)|) \tag{1}$$

With the first estimation, we iterate through every frame in the original signal, and update the noise estimation for that frame using a smoothing value

$$N(m,k) = \beta N(m-1,k) + (1-\beta)P(m,k) \tag{2}$$

where

$$P(m,k) = max[0, |X(m,k)| - |Y(m,k)|] \tag{3}$$

In this equation, P(m,k) represents the single-frame noise estimate. Finally, the updated noise is subtracted from the original signal to yield a clean speech signal:

$$S(m,k) = |X(m,k)| - N(m,k) \tag{4}$$

If we translate this to a gain function:

$$S(m,k) = G(m,k)|X(m,k)| \tag{5}$$

We can account for the SNR between the two signals and say:

$$G(m,k) = 1 - \frac{1}{\sqrt{(SNR(m,k)+1)}} \tag{6}$$

where

$$SNR(m,k) = \beta SNR(m-1,k) + (1-\beta)max[0, S\hat{N}R(m,k)]) \tag{7}$$

and

$$S\hat{N}R(m,k) = \frac{|X(m,k)|}{N(m,k)} - 1 \tag{8}$$

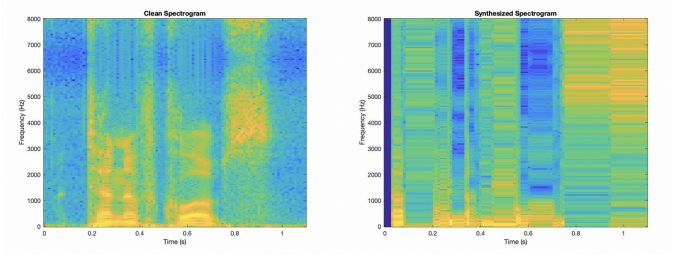

Fig. 2. Spectrogram of clean input speech (left) and synthesized clean speech (right).

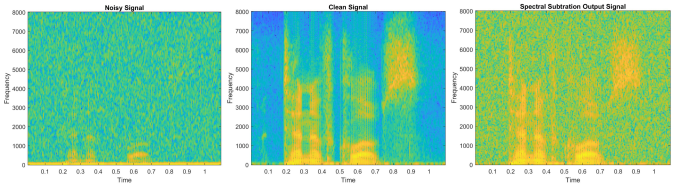

Fig. 3. Results of spectral subtraction in the ideal case of the target signal being exactly the clean signal.

III. Results

A. Speech Synthesis

An example of the result of our speech synthesis implementation can be seen in Figure 2. In looking at the synthesized spectrogram, it can be seen that the fundamental frequencies and the first few harmonics match the original speech spectrogram fairly well, though they largely lack the fine detail and contour of the original speech. However, in the remainder of the frequency spectrum, a significant amount of noise and distortion can be seen in the synthesized spectrogram. In listening to the time-domain version of the synthesized speech, this manifests in the speech sounding incredibly robotic, with the actual speech component sounding barely audible.

B. Spectral Subtraction

To test a baseline version of our spectral subtraction algorithm, we ran it with the noisy signal being a clean speech signal with white noise added, and the "synthesized" signal (the target signal) being the original clean speech. The spectrograms in Figure 3 depict the results when run with a smoothing value beta equal to 0.3. As can be seen, in the noisy signal the speech is highly obscured, but the spectral subtraction algorithm successfully identifies and removes a large portion of the noise.

C. Overall Method

Figure 4 depicts the results of our method when tested a noisy speech signal created by adding white noise to a clean speech signal. As is made evident by the spectrogram of the cleaned signal, the spectral subtraction algorithm currently picks up the noise and distortion added in the synthesis phase of the program more than the speech portion of the spectrogram. While in listening to the synthesized version of the speech signal you could still hear some semblance of the

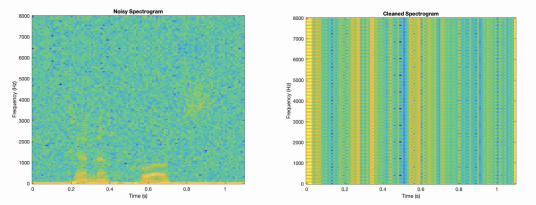

Fig. 4. Spectrogram of noisy input signal (left) and cleaned output signal (right).

speech come through, there is none in the cleaned signal. After spectral subtraction, the result sounds like a combination of buzzing and popping noises.

IV. Conclusion

As it is right now, the speech synthesis algorithm that this method implements is not nearly robust enough to generate a signal that can be used to accurately estimate the noise from the original noisy input. While in the next section we will talk about ways to improve this portion of the method, our opinion is that while the concept poses and interesting solution to speech enhancement, in practice the combination of speech synthesis and spectral subtraction to enhance speech signals is not very practical. In order to synthesize a target signal good enough to use to accurately estimate the noise from an input signal, the synthesized speech would likely already be good enough to use as the output speech already, which would lead to the spectral subtraction step being redundant. Therefore, we think time would be better spent in either significantly improving the speech synthesis section to the point where the synthesized speech can just be used as the output or increasing the effectiveness of spectral subtraction alone. One idea we had for keeping the combine approach somewhat would be to use the phoneme recognition portion of our speech synthesis method as a highly detailed voice activity detector to be used in spectral subtraction.

V. Improvements

While we do think the idea of sticking with this combined approach is overall not worth the effort it would take, we nevertheless have come up with methods of improving our system at least to the point where could generate usable results. The main way in which our method could be improved is by increasing the robustness of the concatenative synthesis. In its current state, our phoneme bank contains only one recorded sample for each phoneme. However, in real speech a person can pronounce the same phoneme in slightly different ways depending on the tone with which they are speaking. The phoneme selection portion of our method would be greatly improved if the phoneme bank contained several different versions of the same phoneme, and the program would select which version of the phoneme to use in synthesis by choosing the nearest neighbor to the input speech.

Another change to our implementation of concatenative synthesis that we think would improve results is to include more smoothing over transitions between frames. While doing overlap-add between individual frames helps, we believe that having even more of a transition, especially between phonemes, would improve the overall contour of the synthesized speech.

References

[1] Lebart, K Boucher, J.-M N Denbigh, P. (2001). A new method based on spectral subtraction for speech dereverberation. Acta Acustica united with Acustica. 87. 359-366.

[2] Vaseghi S.V. (1996) Spectral Subtraction. In: Advanced Signal Processing and Digital Noise Reduction. Vieweg+Teubner Verlag

[3] Lebart, K Boucher, J.-M N Denbigh, P. (2001). A new method based on spectral subtraction for speech dereverberation. Acta Acustica united with Acustica. 87. 359-366.

[4] Neely, Stephen Allen, Jont. (1979). Invertibility of a room impulse response. The Journal of the Acoustical Society of America. 66. 165-169. 10.1121/1.383069.

[5] Lebart, K Boucher, J.-M N Denbigh, P. (2001). A new method based on spectral subtraction for speech dereverberation. Acta Acustica united with Acustica. 87. 359-366.

[6] Han et al. Learning Spectral Mapping for Speech Dereverberation and Denoising. IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 23, NO. 6, JUNE 2015

[7] Xiao et al. EURASIP Journal on Advances in Signal Processing (2016), 2016:4, DOI 10.1186/s13634-015-0300-4

[8] YIN, a fundamental frequency estimator for speech and music, Alain de Cheveigne, Ircam-CNRS, 1 place Igor Stravinsky, 75004 Paris, France