Automatic Volume Leveler for Real Time Speech Applications

Justin M. Bare, Dr. Carol Espy-Wilson, Dr. Tarun Pruthi

***Abstract*— Digital processing of speech signals often has undesirable effects such as amplification or attenuation of parts of the enhanced signal across time and possibly across frequency as compared to the original signal. As a result, the loudness and timbre (or color) of the speech signal may be changed. In real time, corrective processing is complicated because the amount of look-ahead is limited so trivial signal normalization techniques are not an option. In this project, we intend to develop an automatic volume leveler to handle this problem of attenuation/amplification and restore the signal loudness and tilt to a level close to the level of the unprocessed speech signal while also ensuring that clipping does not occur.**

***Index Terms*— automatic volume leveler, real time, speech enhancement, speech processing, volume**

I. Introduction

In real time speech applications, while digital signal processing algorithms may work to enhance the signal in some ways, they may also have negative effects on the signal in other ways. For example, noise reduction processing can perform very well at attenuating the noise in a signal, but it may also inadvertently amplify or attenuate the speech signal (see Figure 1). Accidental amplification or attenuation in this way can cause alterations in the perceived loudness and timbre or coloring of the speech. This is often undesirable if the goal of the processing is to improve the signal-to-noise ratio (SNR) as much as possible or to improve the clarity of the speech. When operating on pre-recorded signals, these issues are not so difficult to fix because the entire signal is available so the noise and speech can be more accurately characterized in order to optimize corrective algorithms. However, in real time scenarios, only the current frame of the signal and past frames of the signal are available at any particular point in time and there is no "ground truth" signal to compare to. This presents a challenge in trying to characterize the noise and speech at a point in time such that the signal can be processed effectively in the near future using these characterizations.

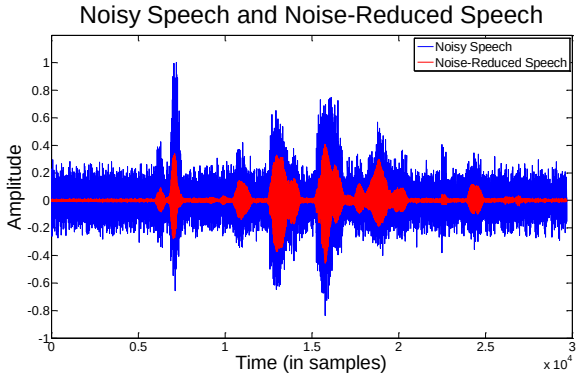

Figure 1. Noisy speech waveform (blue) and waveform of the same signal after noise reduction processing is performed (red). Notice the regions of speech (the high amplitudes) have been attenuated along with the noise.

Therefore, the goal of this project is to develop an automatic volume leveler to correct the unintentional speech amplification or attenuation introduced into the signal by initial processing algorithms, in particular, noise reduction algorithms. The amplitude of the signal in speech regions is to be restored to its original level, but if amplification will lead to clipping, the amplitude will be reduced appropriately to avoid this. Clipping occurs if the floating point values of the signal exceed the range of -1 to 1, and amplification of the signal can produce values outside of this range. The figure below illustrates these goals with the green waveform, which is an example of the output of our volume leveler.

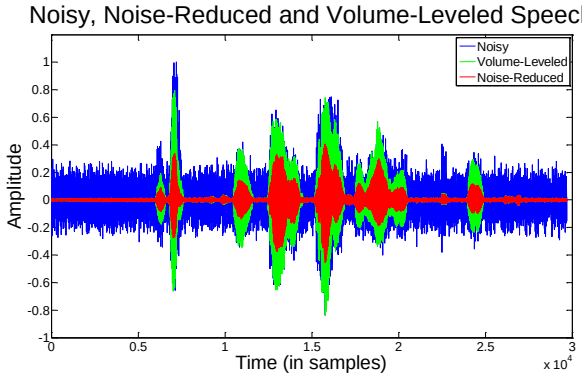

Figure 2. Noisy, volume-leveled, and noise-reduced signals. The green volume-leveled waveform clearly shows restoration of the speech regions as compared to the red waveform of the noise-reduced signal.

Manuscript received August 1, 2011. This material is based upon work supported by the National Science Foundation under Grant No. 1063035.

Justin M. Bare is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: jbare@ umd.edu).

Dr. Carol Espy-WIlson, is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: espy@ umd.edu).

Dr. Tarun Pruthi is an affiliate of the Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (email: t.pruthi@ieee.org).

II. ALGORITHM

Many algorithms were tested, but the most successful one had several key features that made it perform better than the others. Here we will describe these key features and explain the algorithm in the process. To simulate real time constraints while testing in MATLAB, the signal is always divided into frames of length 80 samples and the frames are processed in a loop in order of increasing time. At the end of each iteration, the current frame of samples is multiplied by the calculated scale factor, *K(i)*.

A. Voice Activity Detector

One crucial feature of the algorithm is the use of a Voice Activity Detector (VAD) to determine whether a frame contains speech or not. In the regions where the VAD decides there is no speech (VAD = 0), the signal is always multiplied by 1 to prevent amplifying noise. In regions where there is speech (VAD = 1), the frame is classified as one of three types:

(1) High level speech region

(2) Low level speech region

(3) Noise region

The type of the current frame is determined based on the result of the noise tracking method, which is discussed next.

B. Noise Tracking

During the frames in which VAD = 0, an estimate of the noise level in the processed signal is obtained as

$$mean\_noise = \alpha \cdot mean\_noise + (1-\alpha) \cdot \overline{abs(proc(i))}. \quad (1)$$

Here, *i* is the current frame, *proc* is the processed signal, α is a smoothing factor with value 0.9, and *abs* computes a vector with each element being the absolute value of the corresponding element in the original vector. The bar on the top means that the absolute values of the signal amplitude in the current frame are averaged to get the mean amplitude for the frame. We found the smoothing factor to be necessary to mitigate the effects of instantaneous changes in the noise level during the signal. The equation heavily weights the previous value of *mean_noise* to make sure the estimate changes slowly. This type of smoothing factor appears several times throughout the algorithm. Such a smoothing factor is commonly referred to as the "forgetting factor" – in the sense of how long it takes for the algorithm to forget a previous value by weighting it less significantly.

C. Classifying Regions When VAD = 1

Whenever VAD = 1, the frame is classified based on the following conditions:

(1) High level speech region:

$$\overline{abs(proc(i))} > 3 \cdot mean\_noise \quad (2)$$

(2) Low level speech region:

$$3 \cdot mean\_noise \geq \overline{abs(proc(i))} > 1.5 \cdot mean\_noise \quad (3)$$

(3) Noise region:

$$\overline{abs(proc(i))} \leq 1.5 \cdot mean\_noise \quad (4)$$

The values 1.5 and 3 were determined heuristically by looking at several examples.

The next figure displays a signal with all three of these types of regions present.

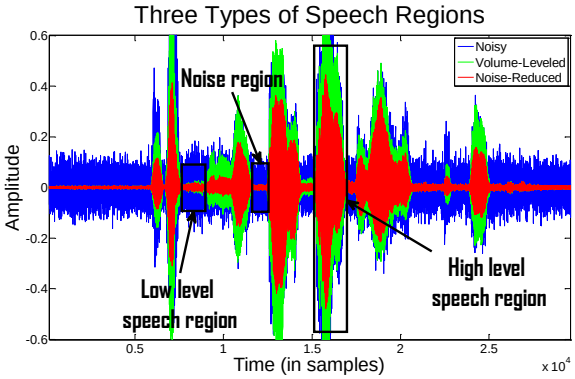

Figure 3. Characteristic examples of the three types of regions, highlighted in rectangles.

D. Calculating the Ratio for the Frame

One important aspect to determine for the algorithm was the basic scaling ratio to use. Several factors were tried such as the ratio of energies in the noisy signal frame to energies in the processed signal frame, or the ratio of the maximum of the noisy signal frame to the maximum of the processed signal frame. When scaling based on the energies, the final scale factor always ended up too large due to the fact that energy is related to the square of the amplitude of the signal. This squaring caused the output signal to be much larger in amplitude than the original signal. Using the ratio of the maximums of each signal caused too much variation in scale factor between frames. The best ratio was found to be the mean of the absolute values of the noisy signal frame divided by the mean of the absolute values of the processed signal frame as in

$$r(i) = \overline{abs(noisy(i))} / \overline{abs(proc(i))} \quad (5)$$

where *r(i)* is the ratio for the current frame and *noisy* is the original signal with noise

E. Calculating the Scaling Factor

Once the type of frame is determined and the ratio is calculated, the scale factor, *K(i)*, for the frame can be calculated. The equations for this are

(1) High level speech region:

$$K(i) = \alpha \cdot prev\_high\_scale + (1-\alpha) \cdot r(i) \quad (6)$$

(2) Low level speech region:

$$K(i) = \beta \cdot K(i-1) + (1-\beta) \cdot r(i) \quad (7)$$

(3) Noise region:

$$K(i) = 1. \quad (8)$$

Here, *prev_high_scale* is the value of *K(j)* where *j* is the most recent previous high level speech frame. This is done to ensure that the algorithm uses the history of scale values already calculated, so that the adaptation of *K(i)* does not start at 1 every time the algorithm encounters a speech region. The value of α is 0.9 and the value of β is 0.99. As these equations show, the purpose of noise tracking and frame classifying is to make sure that only high level speech is significantly amplified. Low level speech is not amplified as much because for low SNR's, these regions sound very noisy if amplified, so therefore the scale factor increases very slowly, with the current ratio receiving little weight compared to the previous scale factor. In addition, low level speech often occurs over very short time periods, so amplifying these sections too much would cause them to sound very discontinuous and artificially loud. Below is graph of the scale factors overlaying the waveforms. Notice how the scale factor changes differently in each type of region.

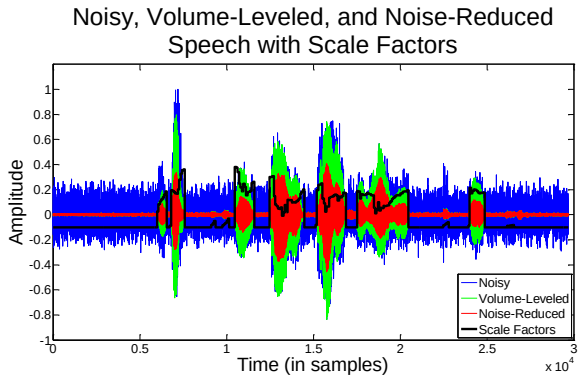

Figure 4. Showing the scale factor during each frame (black curve). The scale factors have been scaled by 1/5 and shifted down for visibility.

F. *Preventing Excessive Amplification*

To make sure the signal is not amplified too much, there are two checks that are performed. First if

$$\max(abs(proc(i))) \cdot K(i) > \max(abs(noisy(i))) \quad (9)$$

then the scale factor is changed by

$$k' = \max(abs(noisy(i))) / [\max(abs(proc(i))) \cdot K(i)] \quad (10)$$

$$K(i) = K(i) \cdot k'. \quad (11)$$

This change is to make sure that the volume-leveled signal does not increase beyond the level of the original signal.

Second, if

$$\max(abs(proc(i))) \cdot K(i) \geq 1 \quad (12)$$

this indicates that the signal is going to clip when it is multiplied by the scale factor, so the scale factor must be adjusted with

$$K(i) = 1/[\max(abs(proc(i))) \cdot \varepsilon]. \quad (13)$$

Here, ε must be a value slightly larger than 1 so that the resulting maximum value in the frame is less than 1. The current implementation has ε = 1.001. This new scale factor becomes a variable called *no_clip_factor* which multiplies all subsequent scale factors in order to ensure consistent amplification and to prevent further clipping.

III. RESULTS

Several measurement techniques are employed to evaluate the performance of the volume leveler. But first, we will discuss the database of speech files used to test the algorithm.

A. *Speech File Database*

The database used for evaluation is made up of 6 different speech utterances, each from one of 6 different speakers, three of which are male and three of which are female. There are clean, non-noisy versions of each of the 6 files and there are also many versions with different types and levels of noise added. There are 22 different noise types and 7 different noise levels. The noise types are widely varying and some examples are babble noise (noise of many people speaking in the background), white noise, and vehicle noise. The noise levels are represented by the SNR and they include -12dB, -3dB, 0dB, 3dB, 6dB, 12dB, and 18dB. So altogether, there are 132 data files (covering all noise types) for each noise level and 42 data files (covering all noise levels) for each noise type.

B. *Amount of Volume Restoration of Speech*

The first method of evaluation involves measuring how close the level of the volume-leveled signal is to the level of the original signal in speech regions in comparison to how close the level of the noise-reduced signal is to the level of the original signal in the same regions. The speech regions are determined using an ideal VAD. Here, ideal VAD means that the non-noisy or clean version of the signal is used to determine exactly when speech is occurring or not occurring. This determination is based on the time-aligned phonetic transcription of the utterance. Frames with a phonetic label are assigned a 1 and the other frames are assigned a 0. In each frame that includes speech, the average of the absolute values of the processed signal is divided by the average of the absolute values of the original noisy signal to obtain the ratio of restoration for that frame. All of these ratios are then averaged across all the speech regions to obtain the level of restoration for the particular file. The figure below displays these results for the volume-leveled signal obtained using our algorithm for each noise type and level. In this case, the

algorithm was implemented with the same ideal VAD as is described above for measuring the results.

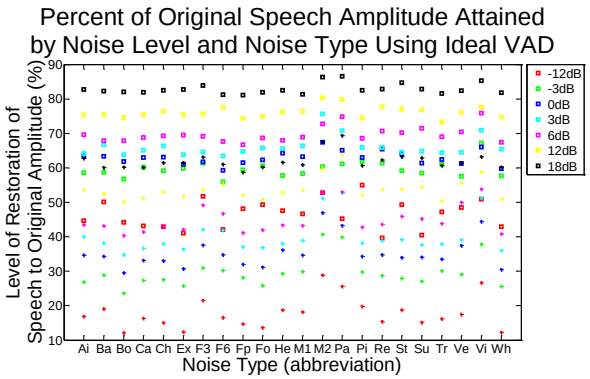

Figure 5. Volume-leveler results are shown as squares and initial noise-reduction results are shown as dots. In all cases, the volume leveler outperforms the noise-reduction algorithm for the same noise type and noise level.

The following figure shows the results in Figure 5, averaged across the 132 data files for each noise level. The difference between the red and green curves below is that the red curve was obtained using the results of volume leveling with a real VAD whereas the green curve was obtained using an ideal VAD, as is described at the beginning of this section. Here real VAD means that a VAD algorithm was used on the original noisy signal to determine when speech was occurring.

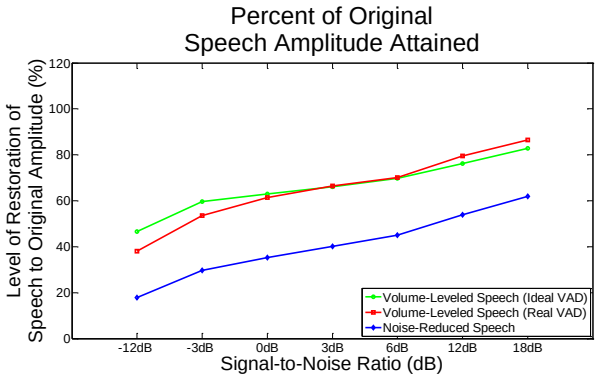

Figure 6. Speech restoration results of the volume leveling algorithm. It is clear that for all SNR's and both VAD's, the algorithm did well to restore speech closer to the original level.

The real VAD often has many errors, either detecting speech when there is no speech or not detecting speech when there is speech. These errors can be significant in some cases. For example, if no speech is detected in a major speech region, the noise estimate will become much higher than it should be due to the way the volume leveling algorithm works to track the noise level. The significance of these errors is apparent in the next set of measurements. Figure 7 shows the degree of difference between the output of the ideal VAD and the real VAD. In this particular case, most of the errors occur as missed detections when the VAD interprets a speech region as non-speech.

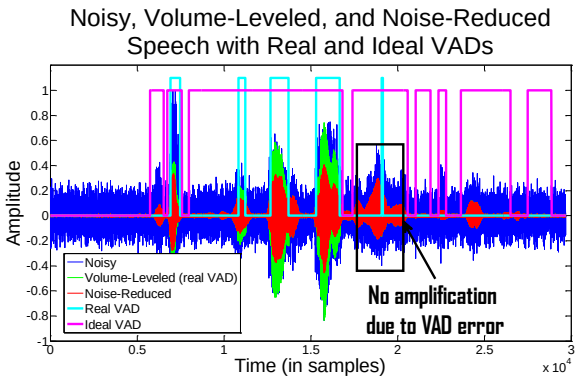

Figure 7. Difference between ideal VAD (magenta) and real VAD (cyan). The real VAD output is scaled by 11/10 for visibility. The green waveform is the volume-leveled signal when real VAD is used. The black rectangle highlights a region when the incorrect VAD decision caused the speech to remain un-restored.

Figure 8 is a comparison of the volume-leveled signals in figures 7 and Figure 4. It is clear from the difference between these signals that the VAD can have a large impact on the success of the algorithm, depending on how accurately the VAD performs.

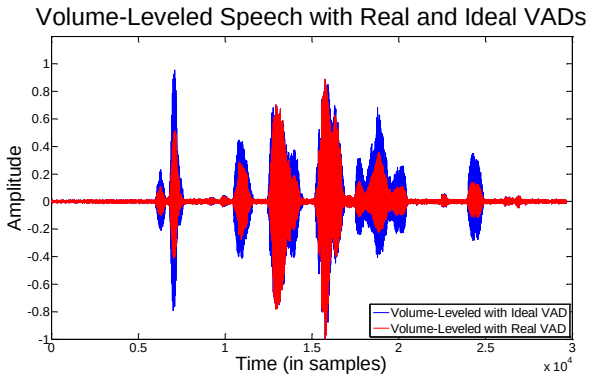

Figure 8. The volume-leveled waveform using the real VAD (red) clearly shows less amplification of many of the speech regions than the waveform resulting from using the ideal VAD (blue).

C. SNRI, TNLR, and NPLR

Techniques for measuring the Signal-to-Noise Ratio improvement (SNRI), total noise level reduction (TNLR) and noise power level reduction (NPRL) were obtained from [1] and are described in full mathematical detail in that recommendation by the ITU.

1) SNRI

SNRI is a measurement that determines how much improvement is obtained in the SNR during speech regions by a particular processing technique. SNRI is computed by subtracting the SNR of the original signal from the SNR of the processed signal. For the evaluation of the volume leveler, the SNRI for the noise-reduced signal was subtracted from the SNRI for the volume-leveled signal in order to determine if the SNR was improved by volume leveling.

2) TNLR

TNLR measures how much the noise level was reduced in non-speech and speech regions of the signal by the processing algorithm. As above, the TNLR for the noise-reduced signal was subtracted from the TNLR for the volume-leveled signal

to determine if the volume leveler increased the noise level. Since TNLR is a negative value, if the TNLR increase is positive, this corresponds to less noise reduction, which means poorer performance in this case.

3) NPLR

NPLR is similar to TNLR, except in this case it only measures the noise reduction in speech regions of the signal. Besides this distinction, it was used in the same way as TNLR for evaluation purposes.

Figures 9 and 10 show the results for these measures for the cases of using the ideal VAD and the real VAD.

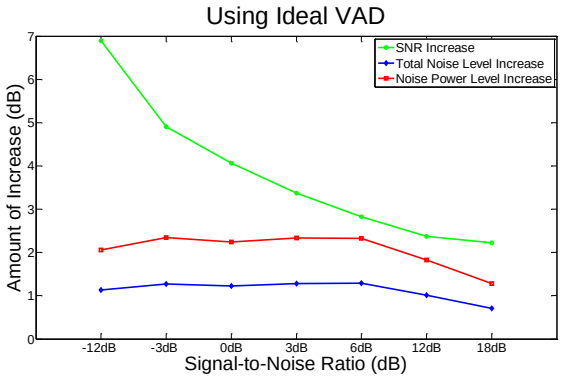

Figure 9. SNR, TNL, and NPL increases for the volume leveler using ideal VAD. These results are positive since the green curve lies above the other two curves.

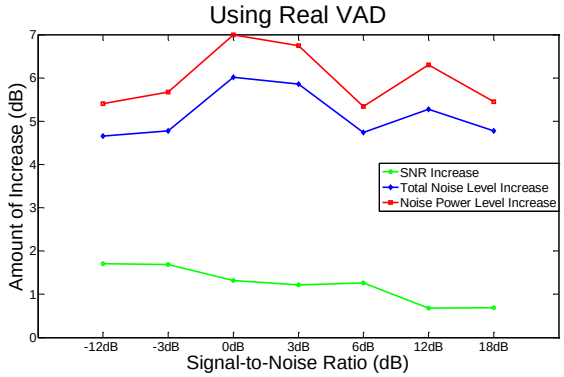

Figure 10. SNR, TNL, and NPL increases for the volume leveler using real VAD. These results are negative since the green curve lies below the other two curves.

Figures 9 and 10 show, the VAD method is a crucial element of the algorithm and its accuracy can significantly affect the performance of the volume leveler. Using the ideal VAD, results were very good in terms of both restoring speech and leaving the noise reduced. Using the real VAD, results were good for restoring speech, but the errors in the VAD caused an increase in noise levels.

IV. Conclusion

We have demonstrated the viability of a volume leveling algorithm for use with real time speech and we have achieved positive results when using an ideal VAD. This demonstrates proof of concept and is a major step towards a usable volume leveler. The current algorithm does not perform as well as necessary for use with a real VAD in real time scenarios, but there are several areas for future work and improvement that could eventually produce a successful algorithm. Most importantly, the algorithm must be less reliant on the VAD, and it would be best if the volume leveler could work without the use of any VAD. Also, the algorithm must eventually be implemented in a fast low level language such as C for effective use in real time. Finally, our algorithm only considered volume, but research should be done on normalizing auditory perceptual loudness as well as incorporating coloring/timbre restoration.

Acknowledgment

Justin M. Bare thanks the National Science Foundation for funding this research, as well as Dr. Carol Espy-Wilson and Dr. Tarun Pruthi for their guidance and support during the summer research.

References

[1] *Objective measures for the characterization of the basic functioning of noise reduction algorithms*, International Telecommunications Union, Recommendation ITU-T G.160, 2008, pp. 29-37.