LabVIEW Implementation of the Dual Channel Noise Reduction Method Exploiting Power Level Differences

by

Ali M. Shtarbanov

Graphical Signal Processing

Karl H. Norian, Ph.D.

April 20, 2014

ABSTRACT

This paper shows a LabVIEW implementation of the dual channel noise reduction and speech enhancement technique proposed in [1]. The approach uses the power level difference of the received signals in two channels as a criterion for noise reduction. A derivation of the algorithm is presented to illustrate the assumptions and the key elements. A detailed explanation of how this algorithm is implemented in LabVIEW is then provided. Due to limitations of the LabVIEW environment, it was not possible to test whether the algorithm worked with real microphones and speakers. Nevertheless, a simulation shows that the algorithm is successful at denoising simulated speech signal buried in noise, where the noise can make up as much as 30% of the amplitude of the speech signal.

INTRODUCTION

Speech enhancement and noise reduction algorithms for single channel signals have existed for many years and have found many applications. Most of these algorithms, however, require knowledge of second order statistics for both the signal and the corrupting noise. This makes them difficult to realize in practice, and also makes some of them environment specific. In more recent years, researches began publishing techniques for dual-channel noise reduction and speech enhancement. Let $x_1[k]$ & $x_2[k]$ denote the signals received by the two microphones, let $s_1[k]$ & $s_2[k]$ denote the speech components of the received signals, and let $n_1[k]$ & $n_2[k]$ denote the noise components of the received signals. Then

$$x_1[k] = s_1[k] + n_1[k] \quad \text{and} \quad x_2[k] = s_2[k] + n_2[k].$$

One of the most basic dual channel noise reduction techniques is the base coherence method, which is based on the idea that the speech components, $s_1[k]$ and $s_2[k]$, are correlated while the noise components, $n_1[k]$ and $n_2[k]$, are uncorrelated [3,4]. The main drawback of this method is that in most practical situations the background noise components, $n_1[k]$ and $n_2[k]$, are highly correlated, and even the same when the two microphones are close together. For such situations, the base coherence method loses its efficiency [2].

A technique that achieves a much higher efficiency, called the Improved Coherence Method, uses the speech-free intervals to calculate the frequency spectrum of the noise, and then uses that result as an estimate for the noise during speech-intervals [5]. Although this approach is powerful, the specific algorithm proposed in [5] results in the presence of a musical noise in the enhanced signal, which is a major disadvantage of the method.

There also exist techniques that are based on the time delay of arrival (TDOA) of input signals at the two channels. One of the most well-known such techniques is the Phase Based Method presented in [6]. The performance of this algorithm is desirable even in the presence of dynamic noise. The major drawback of the method, however, is that it requires an accurate calculation of the time delay between the two signals, which is often very difficult to obtain. Another drawback is poor performance when the two microphones are close to one another [2].

One of the latest techniques for dual microphone noise reduction in speech is the Power Level Difference method presented initially in 2009, and subsequently improved by a different research group in 2012 [1,2]. This technique is free from many of the drawbacks of the aforementioned techniques and is the technique on which the work presented in this paper is based upon. The sections that follow present a complete detailed derivation of the algorithm, implementation in LabVIEW with detailed explanations of each block, simulation results with evaluations, and concluding remarks.

POWER LEVEL DIFFERENCE (PLD) METHOD

Consider a cell phone with a primary microphone on the front side and a secondary microphone on the rear side. Suppose there is a person speaking on that cell phone, who is located in an acoustic environment where the noise field is homogeneous and diffuse. One can easily visualize this situation, but it is difficult to illustrate graphically on paper. Therefore, an equivalent situation is considered here, where a speaker is giving a talk in an environment of a homogeneous and diffuse background noise. The speech and noise are picked up by two microphones – a primary microphone closer to the speaker and a secondary microphone farther from the speaker. Figure 1 is an attempt of illustrating such a situation.

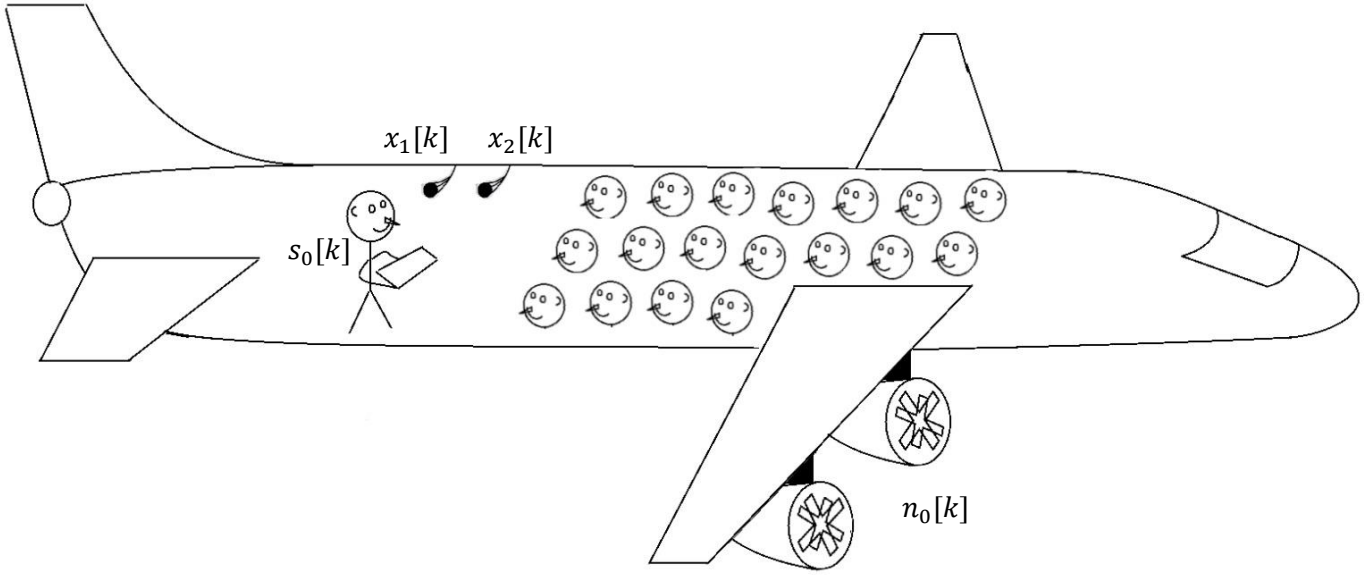


Figure 1. Illustration of an environment with diffuse and homogeneous noise field. The noise component received by both microphones is the same, while the speech component received by the secondary microphone is of lower power than the speech component received by the primary microphone. $s_0[k]$ – speech source (speaker), $n_0[k]$– noise source (engines), $x_1[k]$ – primary input, $x_2[k]$ – secondary input.

The signals received by the primary microphone and the secondary microphone, can be expressed as

$$x_1[k] = s_1[k] + n_1[k] \qquad \text{and} \qquad x_2[k] = s_2[k] + n_2[k]$$

where $s_1[k]$, $s_2[k]$, $n_1[k]$ and $n_2[k]$ are the speech and the noise components *received* by the two microphones. (They are simply attenuated versions of the sources $s_0[k]$ and $n_0[k]$.) The frequency-domain transforms of these expressions are

$$X_1(\Omega) = S_1(\Omega) + N_1(\Omega) \qquad \text{and} \qquad X_2(\Omega) = S_2(\Omega) + N_2(\Omega)$$

In the case where the noise source is far away from the microphones, both microphones will receive the same noise, $N_1(\Omega) = N_2(\Omega) \equiv N(\Omega)$. If one microphone is very close to the speaker and the other one is farther away, then the speech component received by the second microphone, $S_2(\Omega)$, will be weaker compared to speech component received by the primary microphone, $S_2(\Omega)$. Then there exists some transfer function $H(\Omega)$ relating $S_2(\Omega)$ and $S_1(\Omega)$, such that $S_2(\Omega) = H(\Omega)S_1(\Omega)$. The frequency domain expressions can then be written as

$$X_1(\Omega) = S_1(\Omega) + N(\Omega) \qquad \text{and} \qquad X_2(\Omega) = H(\Omega)S_1(\Omega) + N(\Omega)$$

The noise $N(\Omega)$ can be calculated during speech-free intervals. Once the noise PSD is known, the transfer function $H(\Omega)$ can be obtained from the cross power spectral density (CPSD) of the two inputs.

$$\Phi_{X_1X_2} = X_1^* X_2$$

$$\Phi_{X_1X_2} = (S_1^* + N^*)(HS_1 + N^*)$$

$$\Phi_{X_1X_2} = S_1^* S_1 H + N^* S_1 H + S_1^* N + NN$$

$$\Phi_{X_1X_2} = \Phi_{S_1S_1} H + \Phi_{NS_1} H + \Phi_{NS_1} + \Phi_{NN}$$

$$H = \frac{\Phi_{X_1X_2} - \Phi_{NS_1} - \Phi_{NN}}{\Phi_{S_1S_1} + \Phi_{NS_1}}$$

It can be assumed that $S_1(\Omega)$ and $N(\Omega)$ are uncorrelated. Then the transfer function becomes

$$H = \frac{\Phi_{X_1X_2} - \Phi_{NN}}{\Phi_{X_1X_1} - \Phi_{NN}}$$

Or more explicitly

$$H(\Omega) = \frac{\Phi_{X_1X_2}(\Omega) - \Phi_{NN}(\Omega)}{\Phi_{X_1X_1}(\Omega) - \Phi_{NN}(\Omega)}$$

Given the aforementioned acoustic conditions and the nature of the input signals, the system indicated by the block diagram on Figure 2 can be used for noise reduction. In the subsections that follow, a complete description of each component of the system is provided.

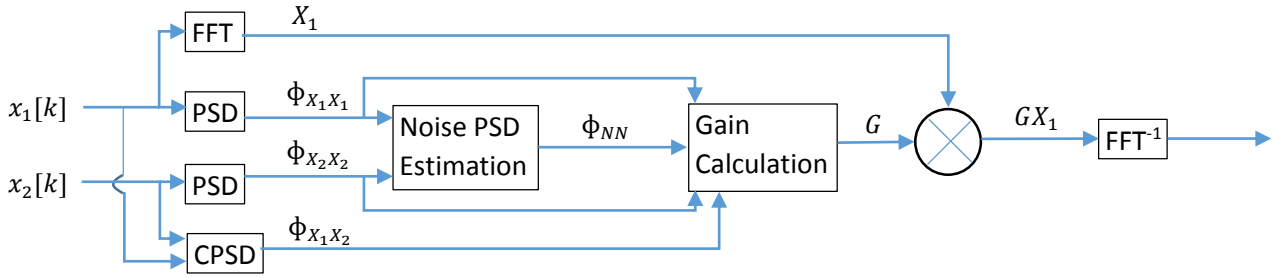


Figure 2. Block diagram of the noise reduction system.

Noise PSD Estimation Block:

In general, the noise will be changing with time. Therefore, every opportunity to calculate the power spectral density (PSD) of the noise must be utilized. Two such opportunities exists:

(I) *When the person is not speaking*, the inputs consist of noise only. Then $\Phi_{NN} = \Phi_{X_2X_2} = \Phi_{X_1X_1}$, and in that situation the PSD of either input would be the PSD of the noise.

(II) *When the person is speaking quietly*, $s_1[k]$ will have a low amplitude and $s_2[k]$ will have even lower amplitude. There exists, therefore, a threshold level below which $s_2[k]$ becomes indistinguishable

from the noise component $n[k]$. Under this condition, $x_1[k] = s_1[k] + n_1[k]$ and $x_2[k] = n_2[k]$. The noise PSD can then be calculated from the secondary input, namely $\Phi_{NN} = \Phi_{X_2X_2}$.

When speech components are present at both inputs, the noise PSD cannot be calculated at that time. However, if the noise does not change very rapidly, then the noise PSD during speech intervals will be approximately equal to the previously calculated value of the noise PSD. Mathematically, this can be expressed as $\Phi_{NN}(\lambda) = \Phi_{NN}(\lambda - 1)$, where λ denotes the frame number.

The aforementioned analysis for calculating the PSD of the noise can be summarized as follows:

$\Phi_{NN} = \Phi_{X_2X_2} = \Phi_{X_1X_1}$	during speech-free intervals
$\Phi_{NN} = \Phi_{X_2X_2}$	during intervals of quiet speech
$\Phi_{NN}(\lambda) = \Phi_{NN}(\lambda - 1)$	during intervals of normal speech

To determine which of the three conditions is occurring at any one time, an indicator and threshold levels are needed. The difference in the PSDs of the two inputs, $\Delta\Phi^{input} = \Phi_{X_1X_1} - \Phi_{X_2X_2}$, can be used as an indicator. During speech-free intervals, $\Delta\Phi^{input} = 0$. During speech intervals, $\Delta\Phi^{input} > 0$. However, because the maximum value of $\Delta\Phi^{input}$ is unbounded, it is impossible to define a threshold level. This can be fixed by taking the normalized difference as an indicator of the condition:

$$\Delta\Phi^{input}_{normalized} = \frac{\Phi_{x_1x_1} - \Phi_{x_2x_2}}{\Phi_{x_1x_1} + \Phi_{x_2x_2}} \text{ will always be less than 1.}$$

In [1], the authors report that for the case of a cell phone with a front-side microphone and a rear-side microphone, when a person is speaking, the difference in the received power between the two microphones is 10dB. This 10dB difference means that the signal received by the secondary microphone has a third of the amplitude of the signal received by the primary microphone. Based on those measurements, thresholds could be established for the value of $\Delta\Phi^{input}_{normalized}$ under each condition:

During speech-free intervals $\Delta\Phi^{input}_{normalized} \approx 0$
During intervals of speech $\Delta\Phi^{input}_{normalized} = \frac{\Phi_{x1x1} - \Phi_{x2x2}}{\Phi_{x1x1} + \Phi_{x2x2}} \approx 0.9$ because $\Phi_{X_2X_2} \ll \Phi_{X_1X_1}$
During quiet speech intervals $\Delta\Phi^{input}_{normalized}$ will be greater than 0 and less than 0.9

To allow for a safety margin, the thresholds can be chosen as 0.2 and 0.8, as proposed in [1]. Then the three conditions become

For speech-free intervals $\Delta\Phi^{input}_{normalized} < 0.2$
For speech intervals $\Delta\Phi^{input}_{normalized} > 0.8$
For quiet speech intervals $\Delta\Phi^{input}_{normalized} \in [0.2, 0.8]$.

The Noise PSD Estimation block can then be mathematically summarized as shown on Figure 3

$$\phi_{NN}(\lambda,\Omega) = \begin{cases} \phi_{X_1X_1}(\lambda,\Omega) = \phi_{X_2X_2}(\lambda,\Omega) & \text{when} \quad \Delta\phi^{input}_{normalized} < 0.2 \\ \phi_{X_2X_2}(\lambda,\Omega) & \text{when} \quad \Delta\phi^{input}_{normalized} \in [0.2, 0.8] \\ \phi_{NN}((\lambda-1),\Omega) & \text{when} \quad \Delta\phi^{input}_{normalized} > 0.8; \end{cases}$$

where Ω is the frequency, λ is the frame index, and $\Delta\phi^{input}_{normalized} = \frac{\phi_{x_1x_1} - \phi_{x_2x_2}}{\phi_{x_1x_1} + \phi_{x_2x_2}}$

Figure 3. Mathematical summary of the Noise PSD estimation block.

Gain Calculation Block:

The purpose of this block is to create a gain function $G(\Omega)$ such that when the primary input $X_1(\Omega)$ is multiplied by this function, the spectral components of the noise that are present in $X_1(\Omega)$ are attenuated and the resulting product is equal to $S_1(\Omega)$. This function can be found as follows:

$$X_1 G = (S_1 + N)G = S_1$$

$$G(\Omega) = \frac{S_1(\Omega)}{S_1(\Omega) + N(\Omega)}$$

This is called the Wiener Filter Equation. When the noise component is 0, then G=1, and when the noise component is large compared to the speech component, then G < 1. In other words, when $X_1(\Omega)$ is multiplied by $G(\Omega)$, only those frequencies, Ω, which contain noise will get attenuated by an amount proportional to the noise, while other frequencies, Ω, that contain no noise will not get attenuated at all. In terms of power spectral densities, the Wiener Filter Equation can be expressed as

$$G(\Omega) = \frac{\phi_{s_1s_1}(\Omega)}{\phi_{s_1s_1}(\Omega) + \phi_{NN}(\Omega) + \phi_{s_1N}(\Omega)}$$

But since the speech and noise components are unrelated, the last term in the denominator is 0. Then the Gain function reduces to

$$G(\Omega) = \frac{\phi_{s_1s_1}(\Omega)}{\phi_{s_1s_1}(\Omega) + \phi_{NN}(\Omega)}$$

To take both inputs into consideration within a single equation, the equation can be modified slightly

$$G = \frac{\phi_{s_1s_1}}{\phi_{s_1s_1} + \phi_{NN}} * \frac{1 - |H|^2}{1 - |H|^2}$$

$$G = \frac{\Phi_{s_1 s_1}(1 - |H|^2)}{\Phi_{s_1 s_1}(1 - |H|^2) + \Phi_{NN}(1 - |H|^2)}$$

where $H = \frac{\Phi_{X_1 X_2} - \Phi_{NN}}{\Phi_{X_1 X_1} - \Phi_{NN}}$ is the transfer function derived in previously in this section.
The numerator of the Gain function turns out to be the difference between the PSD of the primary input signal and the PSD of the secondary input signal. A proof of this statement is in order.

$\Delta\Phi_{input} = \Phi_{X_1 X_1} - \Phi_{X_2 X_2}$ but since S and N are unrelated,

$\Delta\Phi_{input} = \Phi_{S_1 S_1} + \Phi_{NN} - \Phi_{S_2 S_2} - \Phi_{NN}$

$\Delta\Phi_{input} = \Phi_{S_1 S_1} - \Phi_{S_2 S_2}$

$\Delta\Phi_{input} = \Phi_{S_1 S_1} - |H|^2 \Phi_{s_1 s_1}$

$\Delta\Phi_{input} = \Phi_{s_1 s_1}(1 - |H|^2)$

The Gain function can then be expressed as

$$G = \frac{\Delta\Phi_{input}}{\Delta\Phi_{input} + \Phi_{NN}(1 - |H|^2)}$$

$$G = \frac{\Phi_{X_1 X_1} - \Phi_{X_2 X_2}}{\Phi_{X_1 X_1} - \Phi_{X_2 X_2} + \Phi_{NN}(1 - |H|^2)}$$

$$G = \frac{\Phi_{X_1 X_1} - \Phi_{X_2 X_2}}{\Phi_{X_1 X_1} - \Phi_{X_2 X_2} + \Phi_{NN} - \Phi_{NN}|H|^2}$$

A suggestion is made in [1] that the noise PSD be multiplied by an overestimation factor of 4. Then

$$G = \frac{\Phi_{X_1 X_1} - \Phi_{X_2 X_2}}{\Phi_{X_1 X_1} - \Phi_{X_2 X_2} + 4\Phi_{NN} - 4\Phi_{NN}|H|^2}$$

The Gain Calculation block can then be mathematically summarized as shown on Figure 4.

$$G = \frac{\Phi_{X_1 X_1} - \Phi_{X_2 X_2}}{\Phi_{X_1 X_1} - \Phi_{X_2 X_2} + 4\Phi_{NN} - 4\Phi_{NN}|H|^2} \qquad \text{where} \qquad H = \frac{\Phi_{X_1 X_2} - \Phi_{NN}}{\Phi_{X_1 X_1} - \Phi_{NN}}$$

Figure 4. Mathematical summary of the Noise PSD estimation block.

LabVIEW IMPLEMENTATION

The layout of the LabVIEW code is identical to the block diagram from Figure 2. One part that is not shown on the block diagram but is present in the LabVIEW code is a signal generation block where the input signals are simulated/generated.

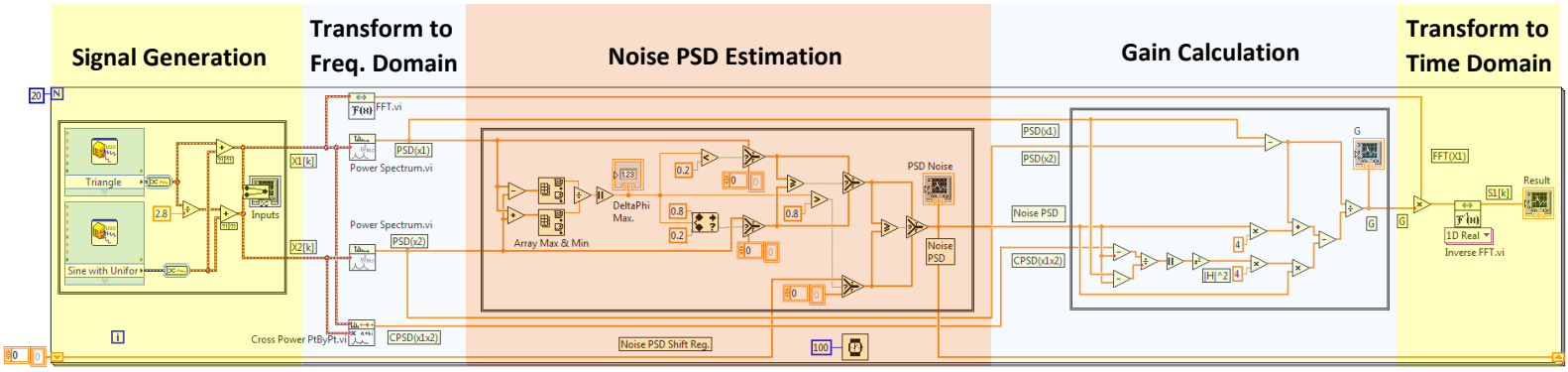


Figure 4. Implementation of the block diagram from Figure 2 into LabVIEW, and the five main components of the VI: Signal generation, Time-domain to frequency-domain transformation, Noise PSD Estimation, Gain calculation, Frequency-domain to time-domain transformation.

Signal Generation Block:

The two main assumptions about the acoustic environment required for the system to work were mention in an earlier section of this work. A brief summary of those assumptions is provided here again.

(1) The background noise field is homogeneous and diffuse, therefore, the noise components received by the two microphones are identical.

(2) The speech component of the signal received by the secondary microphone is about 10dB weaker compared to the speech component of the signal received by the primary microphone.

To simulate the primary and the secondary input signals, such that they satisfy the aforementioned acoustic conditions, the strategy described in this paragraph was implemented as shown on Figure 5. The signal on Channel 1, labeled X1[k], is made up of a wave (either triangular or sinusoidal) of unit amplitude and random noise of amplitude 0.25. The signal on Channel 2 is made up of an attenuated version of the same clean wave added to the same random noise of amplitude 0.25. The crucial point to be noted is that on both channels, the same noise is added to a *clean* wave, thereby ensuring assumption (1) is satisfied. Figure 5b shows the generated waveforms for the case of a triangle wave input.

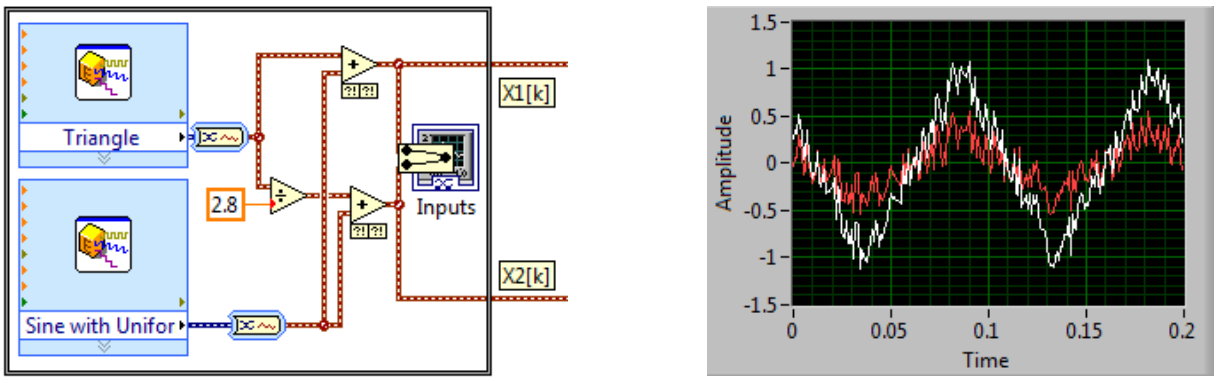


Figure 5. a) LabVIEW implementation of the Signal Generation Block and b) the corresponding waveforms on the two channels. The white-colored wave is the primary input $x_1[k]$ and the red-colored wave is the secondary input $x_2[k]$.

Frequency Domain Transformation Block:

The generated signals $x_1[k]$ and $x_2[k]$ are transformed to the frequency domain as shown on Figure 6. FFT.vi takes the Fast Fourier Transform of $x_1[k]$ and outputs $X_1(\Omega)$, which is complex-valued. The two virtual instruments, both titled Power Spectrum.vi, calculate the power spectral density (PSD) of the signals $x_1[k]$ and $x_2[k]$ and output $\Phi_{X_1X_1}$ and $\Phi_{X_2X_2}$, which on Figure 6 are labeled as PSD(x1) and PSD(x2), respectively. The Power Spectrum LabVIEW function takes the FFT of the input signal, multiplies the resulting complex result by its complex conjugate, then squares that product, and divides it by the number of samples in the frame. The virtual instrument titled Cross Power.vi calculates the cross power spectral density (CPSD) of $x_1[k]$ and $x_2[k]$ and yields $\Phi_{X_1X_2}$, which on the figure is labeled as CPSD(x1x2).

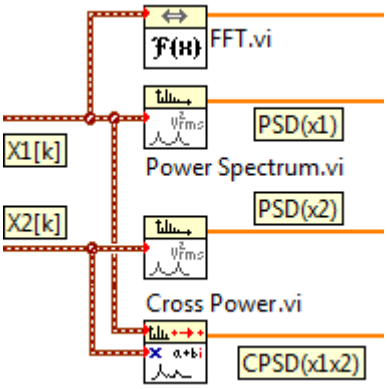
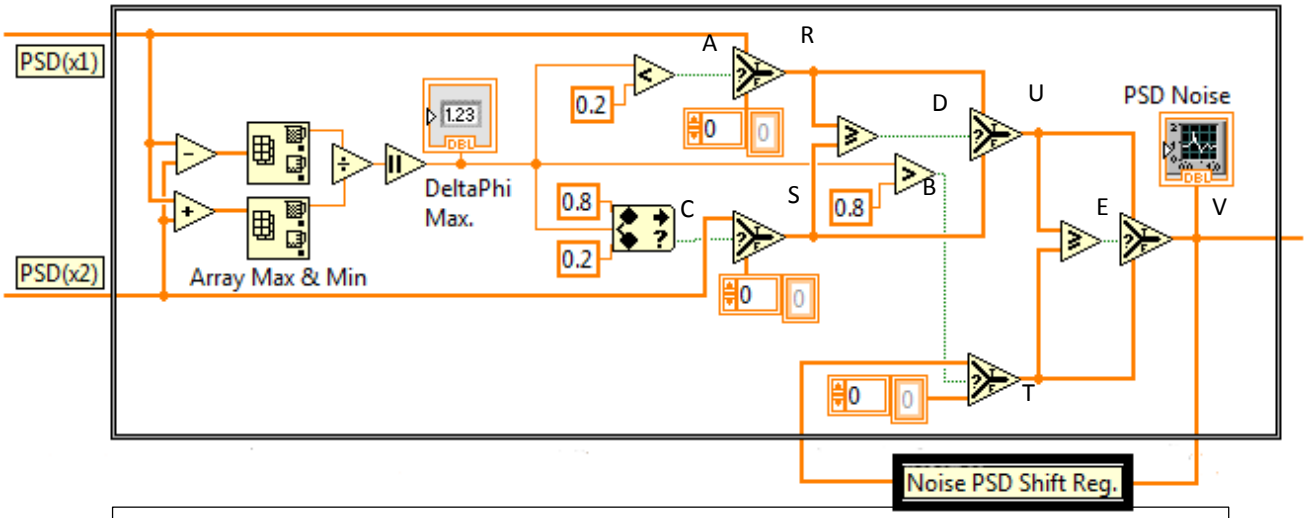


Figure 6. Frequency-domain transformation block.

Noise PSD Estimation Block:



$$\Phi_{NN}(\lambda,\Omega) = \begin{cases} \Phi_{X_1X_1}(\lambda,\Omega) = \Phi_{X_2X_2}(\lambda,\Omega) & \text{when } \Delta\Phi^{input}_{normalized} < 0.2 \\ \Phi_{X_2X_2}(\lambda,\Omega) & \text{when } \Delta\Phi^{input}_{normalized} \in [0.2, 0.8] \\ \Phi_{NN}((\lambda-1),\Omega) & \text{when } \Delta\Phi^{input}_{normalized} > 0.8; \end{cases}$$

where Ω is the frequency, λ is the frame index, and $\Delta\Phi^{input}_{normalized} = \frac{\Phi_{x_1x_1} - \Phi_{x_2x_2}}{\Phi_{x_1x_1} + \Phi_{x_2x_2}}$

Figure 7. (a)Noise PSD Estimation Block and (b) the corresponding equations. The labels A,B,C,D,E denote *Boolean variables*, while the labels R,S,T,U,V denote *array variables*, carried by the corresponding wires.

Figure 7a shows the LabVIEW implementation of the noise PSD estimation block, which is nothing more than the implementation of the conditional equation from Figure 7b. This current section describes in excruciating detail the implementation of the block and the purpose of every function within the block.

PSD(x1) and PSD(x2) are the two input arrays that enter the block. They come from the Frequency Domain Transformation block from Figure 6. The leftmost side of the Noise PSD Estimation Block from Figure 7a computes

$$\Delta\phi_{normalized}^{input} = \frac{\phi_{x_1x_1} - \phi_{x_2x_2}}{\phi_{x_1x_1} + \phi_{x_2x_2}}$$

The numerator is computed by subtracting PSD(x2) from PSD(x1), element by element, and the denominator is computed by adding PSD(x2) to PSD(x1), element by element. Since the maximum value of the threshold, $\Delta\phi_{normalized}^{input}$, is sought, the maximum values of the sum and the difference are extracted, the division is performed, and the absolute value of the result is taken. This yields DeltaPhi Max., as indicated on the Figure 7a, which is the maximum value of $\Delta\phi_{normalized}^{input}$.

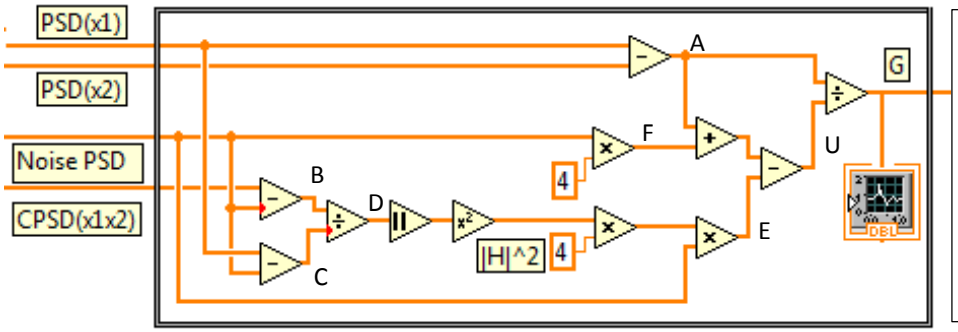
Once DeltaPhi Max is known, the program then proceeds to compute $\phi_{NN}(\lambda, \Omega)$. The program determines whether DeltaPhi Max is less than 0.2, greater than 0.8, or between 0.2 and 0.8. The Boolean results of these comparisons are the wires labeled A, B, and C, respectively. Since DeltaPhi Max is bounded between 0 and 1, and since the three comparison operations are mutually exclusive, one of the three Boolean results will always be true and the remaining two will always be false: If A=TRUE, then B=C=FALSE; if B=TRUE, then A=C=FALSE; if C=TRUE, then A=B=FALSE.

The Boolean results A,B, and C then each enter a Select Function, whose symbolic representation is a triangle with three inputs (labeled 'T','F', and '?') and one output. A Select Function returns the value wired to the 'T' input or 'F' input, depending on the Boolean value wired to the '?' input. If '?'=TRUE, the function returns the value wired to 'T'. If '?'=FALSE, the function returns the value wired to 'F'.

The three Select Functions that are controlled by the Boolean values A, B, and C all have their 'F' input wired to an array of zeroes. Therefore, since only one of the Boolean controls A, B, C will be TRUE at any one time, then only one of the outputs R, S, T can be nonzero at any one time. When A=TRUE, then R=PSD(x1), S=0, T=0; when B=TRUE, then R=0, S=PSD(x2), T=0; and when C=TRUE, then R=0, S=0, T={PSD(noise) from the previous iteration}. In order to have T={PSD(noise) from previous iteration} when C=TRUE, the PSD(noise) is connected to a shift register and the output of the shift register is connected to the 'T' input of the Select Function.

At this stage of the code, the program knows that only one of the three outputs R, S, T is nonzero, but it does not know which one. In order to determine the nonzero output, R, S, and T must be compared against each other. There is a number of equivalent ways of accomplishing this comparison. The implementation shown on Figure 6a shows one of the possible methods of comparison. R and S are compared first to yield U, which is the larger of R and S. However, there is a slight caveat when performing the comparison between R and S. There exists the possibility that R and S are both zero, in which case R=S. It is therefore necessary to use the Greater Than Or Equal To Function rather than the Greater Than Function. If R is greater than or equal to S, then U=R; and if R is less than S, then U=S. The last comparison to be made is that between U and T. There is another caveat in this comparison that is even more difficult to detect. When the program is running, U and T can never be both 0; thus U can never be equal to T. However, only during the first iteration of the program, there exists a condition (albeit highly improbable) where both T and U are 0. Therefore, the Greater Than Or Equal To Function is used for this comparison rather than the Greater Than Function. The output is PSD(noise).

Gain Calculation Block:



$$G = \frac{\Phi_{X_1X_1} - \Phi_{X_2X_2}}{\Phi_{X_1X_1} - \Phi_{X_2X_2} + 4\Phi_{NN} - 4\Phi_{NN}|H|^2}$$

where $H = \frac{\Phi_{X_1X_2} - \Phi_{NN}}{\Phi_{X_1X_1} - \Phi_{NN}}$

Figure 8. (a)Gain calculation block and (b) the corresponding equation

The Gain Calculation Block shown on Figure 8a is simply an implementation in LabVIEW of the Gain equation from Figure 8b. The inputs to this block are PSD(x1), PSD(x2), PSD(noise), and CPSD(x1x2), and the output of the block is the Gain function G. PSD(x2) is subtracted from PSD(x1) and the result, labeled A, is the numerator of the Gain equation. The function H is then computed. B and C indicate the numerator [CPSD(x1x2) - PSD(noise)] and denominator [PSD(x1) - PSD(noise)] of H, respectively. B and C are then connected to a Divide Function, and the resulting output, labeled D, is the function H.

Once H has been determined, the denominator of the Gain function is then computed as follows: The absolute value of H is obtained and is then squared to yield |H|^2 as labeled on the figure. This is then multiplied by 4 and by PSD(noise) to yield the last term in the denominator of the Gain function – labeled E on the figure. The other three terms of the denominator are labeled PSD(x1), PSD(x2), and F on Figure 8a. The terms are then added/subtracted appropriately to yield the denominator, labeled U. The numerator A is then divided by the denominator U and that yields the final result G.

Time-Domain Transformation Block:

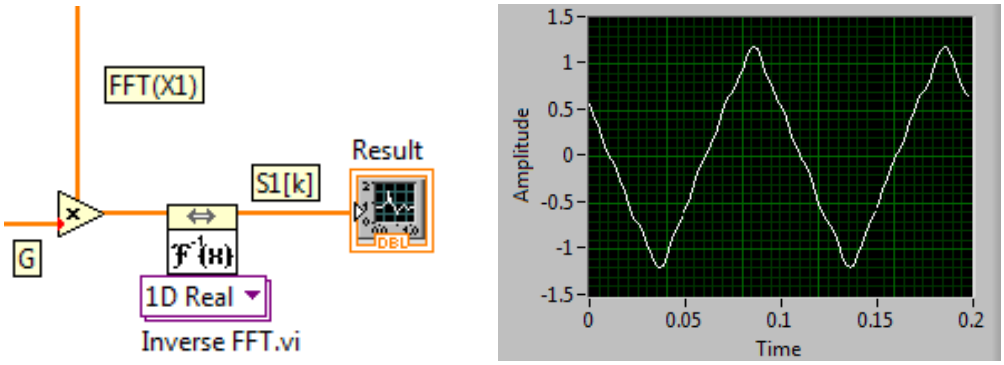


Figure 9. (a) Time-Domain transformation block and (b) the resulting cleaned output signal.

In Figure 9, FFT(x1), is multiplied by the output of the gain calculation block, labeled G. FFT(x1) is the Fourier transform of the primary input. In general, this signal contains a speech component and a noise component. G is the Gain function, which is the output of the Gain Calculation block. The Gain function is simply a filter designed specifically to attenuate the noise content in the input signal and preserve the speech content of the input signal unchanged. G is equal to 1 at those frequency components that contain

no noise and is equal to less than 1 at those components that contain noise. Therefore, the product G*FFT(x1) should approximately equal FFT(S1) – the Fourier transform of the speech component of the primary input signal. This result is then sent through an Inverse FFT function to convert it back to the time-domain. The output, labeled S1[k], is approximately equal to the speech component of the primary input signal.

EVALUATION

The LabVIEW programming environment has a limitation, which did not allow the system to be tested with a set of real microphones and real speakers. In particular, there is a significant delay between the completion of one iteration of a loop and the beginning of the next iteration of the loop. This delay is not the delay associated with the completion of one iteration, it is the delay *between* iterations. This LabVIEW limitation/bug was verified by designing a very simple VI consisting of a Simulate Signal Function and an Audio Output Function, placed inside a while loop. The simulated signal was a sine wave engineered to have 0 phase offset at the start of an iteration and 0 phase offset at the end of the iteration. By ensuring that the phase offsets were 0, it was expected that when this simple VI is run, it would generate a continuous sine wave and therefore a smooth audio tone at the speakers. This was not the case, however. Instead of a smooth and continuous audio signal at the output, the program generated an audio signal that was discontinuous. Moreover, a crackling noise was audible between iterations.

Due to the aforementioned bug/limitation with LabVIEW, the system was tested using simulated input signals with simulated background noise, as described in the previous section. The results of the simulation were pleasing and matched the theoretically predicted expectation.

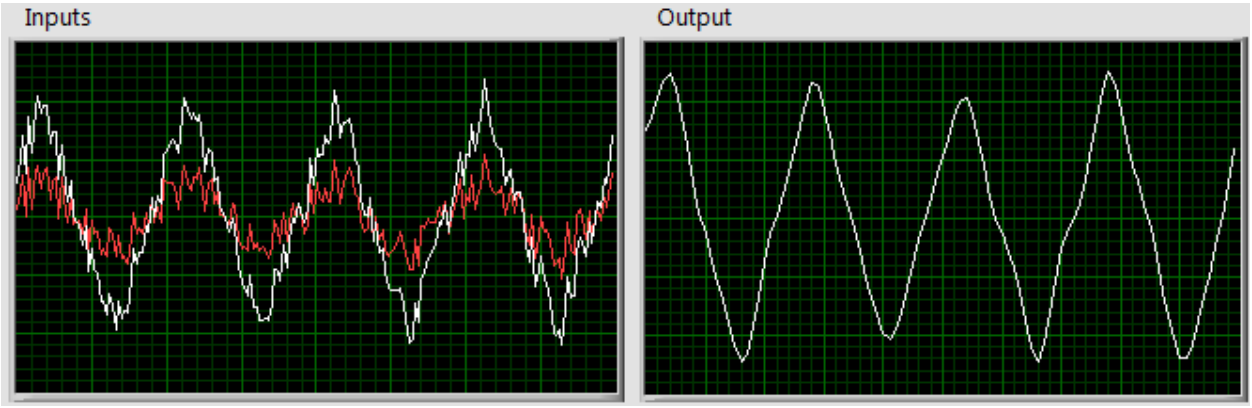


Figure 10. (a) Dual channel input signals (triangle wave with noise). (b) Output signal.

Figure 10 shows the noisy input on the two channels and the cleaned-up output signal. For the simulation on Figure 10, the speech component of the input signal was assumed to be a triangle wave and the noise component of the input signal was randomly generated noise. The cleaned output signal closely resembles a triangle wave. It should be noted that the system preserves the sharp peaks of the triangle wave. Because of this preservation of high frequency components, the dual channel noise reduction system presented in this work is far superior to most noise reduction systems on the market

today, which rely heavily on low pass filtering, and therefore suppress high frequency components of the speech signal.

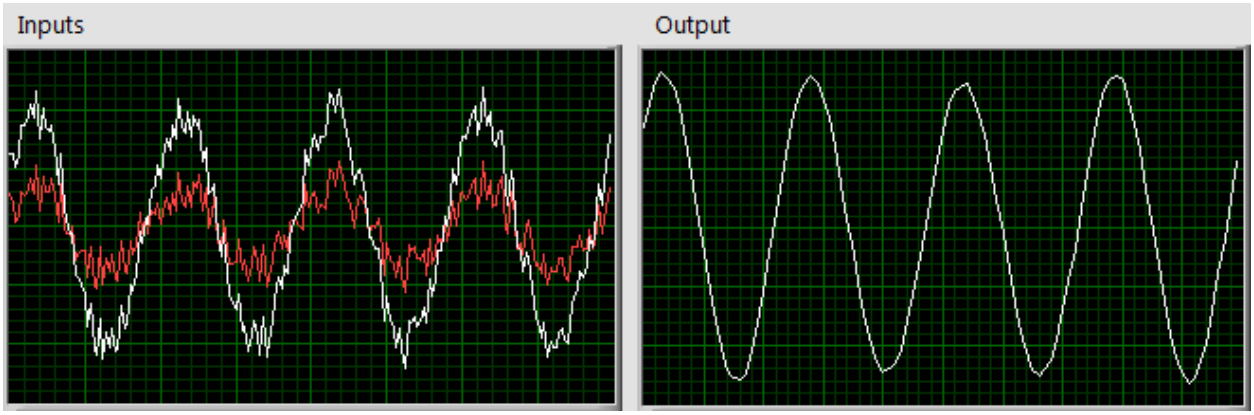


Figure 11. (a) Dual channel input signals (sine wave with noise). (b) Output signal.

Figure 11 shows the input and output for a different simulation, where a sine wave was used to represent the speech component of the input. Both, the sine wave simulation and the triangle wave simulation, were performed with 200 samples using a rate of 1000 samples per second. It was also observed that for random noise whose amplitude is up to 30% of the amplitude of the speech signal, the system is successful at cleaning the noise. The sine wave simulation and the triangle wave simulation were the only two simulations that could be performed using the standard simulation tools in LabVIEW. Performing simulations using waveforms that more closely resemble human speech would have required exceedingly long time to simply engineer the waveforms, which was beyond the scope of this work. Even though the sine wave and the triangle wave do not realistically resemble human voice, they are a good indicator that the system is successfully able to suppress the background noise.

CONCLUSION

This work rederived the algorithm for the dual channel noise reduction system using power level differences proposed in [1], showed an implementation of the system in LabVIEW, and tested the performance of the system. The central components of the algorithm were noise PSD estimation algorithm and spectral gain calculation algorithm. The LabVIEW implementation consisted of a signal generation block, time domain to frequency domain transformation functions, noise PSD estimation block, gain calculation block, and frequency domain to time domain transformation function. Due to limitations of the LabVIEW environment, it was not possible to test the system using real microphones and speakers, and therefore simulation remained the only alternative option. Two simulations were performed altogether – one featuring a triangle wave as the speech component of the input signal, and another featuring a sine wave as the speech component of the input signal. The results of the simulations were successful and the system was able to successfully clean up noise whose magnitude was up to 30% of the magnitude of the primary input signal. Moreover the dual channel noise reduction system is far superior to many noise reduction system on the market today that rely on low pass filters, because it preserves high frequency components and attenuates only the noise – regardless of whether that noise is of low or of high frequency.

A number of improvements could be made to the system to achieve an even greater success with noise reduction. The noise PSD estimation block features two estimated threshold levels (0.2 and 0.8) which were assumed to apply for environments. These threshold levels would be slightly different for different environments, and they can be designed to be user adjustable. Moreover, there could be some logic in the implementation block that makes a more accurate estimation of the thresholds and updates them continuously. Another improvement would be to use a recursive smoothing technique, which was suggested by the authors of [1], where the estimated noise PSD during the next consecutive iteration of the loop is a function of the present iteration of the loop. This would be most useful for the case of environments where the background noise resembles a Dirac delta function (e.g. noise from a jackhammer). Another improvement would be to use a third or even a fourth microphone that is even farther from the speaker. This would allow a far more accurate calculation for the PSD of the background noise. All of these improvements would yield better results, but they would increase the complexity and the cost of the system, which may not be desirable.

REFERENCES

[1] M. Jeub, C Hergoltz, C Nelke, C. Beaugeant, and P. Vary, "Noise Reduction for Dual-Microphone Mobile Phones Exploiting Power Level Differences," in *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2012.

[2] N. Youse□an, A. Akbari, and M. Rahmani, "Using power level difference for near field dual-microphone speech enhancement," *Applied Acoustics*, vol. 70, pp. 1412– 1421, 2009.

[3] Allen JB, Berkley DA, Blauert J. "Multi Microphone Signal Processing Technique to Remove Reverberation from Speech Signals," J Acoust Soc of Am 1977.

[4] Le R, Bouquin A, Faucon G. "Using the coherence function for noise reduction," in Processings of the IEEE on communications, speech and vision, vol. 139, 1992.

[5] Bouquin-Jeanes RL, Asirani AA, Faucon G. Enhancement of speech degraded by coherent and incoherent noise using a cross-spectral estimator. IEEE Trans Speech Audio Process 1997.

[6] Aarabi P, Shi G. Phase-based dual-microphone robust speech enhancement. IEEE Trans Syste, Man Cybern 2004.