RTDSP Project

Contents

[Program Function 2](#_Toc351917544)

[Evaluation of different enhancements 2](#_Toc351917545)

[Enhancement 1 2](#_Toc351917546)

[Enhancement 2 2](#_Toc351917547)

[Additional Enhancements 2](#_Toc351917548)

[First proposed enhancement 3](#_Toc351917549)

[Description 3](#_Toc351917550)

[Results 3](#_Toc351917551)

[Second proposed enhancement (VAD) 3](#_Toc351917552)

[Implementation and description 3](#_Toc351917553)

[Results 4](#_Toc351917554)

[Future Improvements 5](#_Toc351917555)

[Works Cited 5](#_Toc351917556)

[Appendix 6](#_Toc351917557)

[Code snippets 6](#_Toc351917558)

# Program Function

*Your report should give formulae that specify precisely what quantities are calculated in your program.*

The program works by assigning the 4 buffers as pointers to allow for efficient switching later on (by simply switching the pointers rather than copying each element into each other – another method which would have also worked would have been to implement a two-dimensional array [4][FFTLEN] and setup the first index in a way to act circular).

Another second important structure of the program is the use of pre-processor directives to enable or disable different enhancements. This was chosen instead of if statements in order to avoid unnecessary computations to be done at runtime – it was found that if frame processing time was too lengthy crackling could be heard at the output thus explaining the need for code efficiency.

Another enhancement included in the program was to use the symmetry of the Fast Fourier Transform to halve the number of computations. Thus when operating in frequency domain all for loops were of the form: for (k=0;k<FFTLEN/2;k++) and before converting back to time domain the remaining of the samples were copied back in a symmetric manner.

# Evaluation of different enhancements

*your report should make clear what compromises you make in choosing your final algorithm.*

*You must also try to give explanations of your understanding of why the enhancements work (or do not work) better than the simple algorithm*

## Enhancement 1

The aim of the first enhancement is to tackle the issue that components at in each frequency bin greatly vary. For example each frequency bin can be assumed to have Gaussian distribution, meaning each element can be represented[[1]](#footnote-1) by, where and vary for each frequency bin. Combining this concept with the equation used to estimate the noise for each noise buffer estimate,, it can be seen that will only contain the low value spectrum components of noise (i.e. the values located in the lower part of the Gaussian distribution), explaining the need for a high alpha. While a high alpha somewhat fixes these problems there is the issue that when the mean or variance is different a high alpha will not fix this issue as the various frequency bins would need to be increased by a different value of alpha to represent the average value of noise (used to remove it from the signal). For example it can be assumed that the captured noise is (approximately) represented[[2]](#footnote-3) by showing how if and changes for each sample, simply increasing by alpha will not work. As the average is what is sought for reading a moving average would be more effective. This is the idea behind the first enhancement.

A moving-average (or low pass filter) can be implemented by a first order exponentially weighted moving average (EWMA [1], which is a type of IIR filter) of the form by choosing an appropriate value. To choose we can use with being the time constant essentially representing the amount of memory (i.e. how long the system would, to an extent, forget this past input). A time constant of 0.85 was chosen by using a time constant of giving .

With this enhancement alpha was able to be reduced from around 30 to 2 in addition to noise being better removed, due to being estimated more accurately.

## Enhancement 2

The second proposed enhancement is very similar to the previous one with the change that the low pass filter is done not in amplitude domain but in power domain, effectively squaring the amplitudes, meaning the equation thus becomes. This one was found to be more effective for the reasons that sound is heard by humans in the power, meaning that the estimate of noise corresponds more to what humans will hear it as.

A more mathematical explanation is that squaring all values means that larger values have a larger weighting. Thus the noise estimate will be more responsive to amplitude increases and will generally return a higher estimate of noise than the previous enhancement.

## Enhancement 7

This enhancement relies on changing the size of the FFT length used for the overlap add algorithm:

|  |
| --- |
| **#define** **FFTLEN 256** |

Decreasing the FFT length resulted in the signal sounding tinnier with musical noise added on top. The reason for this is that by decreasing the FFT length means frequencies are being quantized even more which will transform back into signals with less fundamental frequency components. For example when taking 256 samples the first frequency bin represents all frequencies of the signal from 0 to 31.25 Hz whereas taking 128 samples the first frequency bin represents frequencies from 0 to 62.5 Hz and in the extreme case of having a length of 16 the first frequency bin would represent frequencies from 0 to 500Hz. It is easy to see why this would create problems and why a higher FFT length is more desired as sound is perceived logarithmically and low FFT length cause the hearing experience to be worse.

Increasing the FFT length caused the lag between input and output to be increased but more importantly it quantizes the sound in sound domain in a similar manner to what happened in frequency domain when decreasing the FFT length.

Unfortunately both good time and frequency resolutions are sought for and in this situation 256 was found to be the best compromise available.

# Additional Enhancements

Two enhancements were added, both variations on the 6th suggested enhancement, which was to deliberately increase alpha for low frequency bins, thus overestimating the noise collected for in those bins.

## First proposed enhancement

### Description

The first modification involved increasing alpha as it moved away from frequency bins containing the fundamental frequency of human speech, which are from around 80 Hz to 260Hz [2]. Due to speech harmonics being important for speech intelligibility [3], alpha was chosen to be less exaggerated at higher frequencies. From this the coefficients were generated using the Code Snippet 1, found in the appendix, and can be seen in **Figure 1**. The idea behind this implementation is that if frequencies which do not contain speech frequencies have noise a bit more aggressively removed then speech intelligibility may be improved. Thus this enhancement is the compromise between creating a pass band filter only allowing the fundamental speech frequencies through (which would not work due to removing all harmonics).



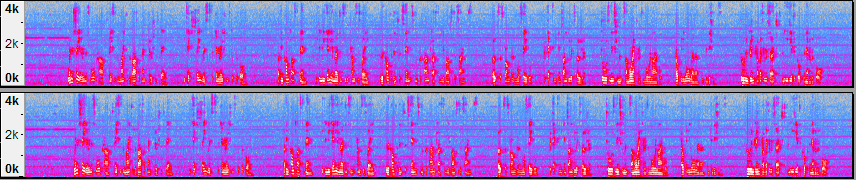
Figure 1 Visual plot of alpha coefficients use to exaggerate the noise at certain frequencies. The reason for the symmetric coefficients is due to the input of the FFT being symmetric. All coefficients are normalized thus can be multiplied with alpha to obtain the desired effect.

From the coefficients which we can store in an array called alpha\_coefs[] we can then use them in the following way when defining the noise estimate for the currently processed frame:

|  |
| --- |
| N[k] = alpha\*alpha\_coefs **[**k**]**\***min**(**min**(M1**[**k**]**,M2**[**k**]**),**min**(M3**[**k**]**,M4**[**k**]**)); |

### Results

**Figure 2** shows the spectrogram with and without the proposed enhancement. While it is difficult to see, the speech components (mainly in red and blue) are conserved across both tests. However the background noise (identifiable by the constant and across time blue components) can be seen to be less present (the blue shades are overall less intense). Listening to these samples does reveal that noise is less present.



With enhancement

Without enhancement

0

40

Time

Frequency

Frequency

Figure 2 Spectrogram of the file lynx1.wav with and without the enhancement. The more intense the colour indicates a high presence of the frequency whereas a grey/blue colour implies the frequency components at the frequency are very low.

Other alpha amplification values were tested and higher values, as expected, also “muffled out” the voice decreasing intelligibility whereas lower values simply did not help with the implementation. Additionally it can be noted that this technique is ineffective at improving performance when the noise is at similar frequencies to speech.

## Second proposed enhancement (VAD)

### Implementation and description

The second implementation functioned as basic voice activity detection algorithm. During periods where voice was not detected the output was highly attenuated thus removing a majority of noise during non-speech periods. While this implementation does not increase sound intelligibility it does increase the listening experience – for example removing the occasional noises found in the “factory” files[[3]](#footnote-4). Additionally this algorithm is not strictly a voice detection algorithm, it is more of a constant background noise detector. Had more time been available methods to improve speech intelligibility would have been possible by building upon this first effort.

This enhancement first relied on calculating the SNR of the signal across all frequencies. Thus signal power was calculated by S\_Power += X[k]\*X[k] (done inside an if loop to add up from each frequency) and recursively looping through each element and noise was similarly calculated by N\_Power += N[k]\*N[k]/(alpha\*alpha\*lpfcoef[k]\*lpfcoef[k]) (the division by alpha\*alpha\*lpfcoef[k]\*lpfcoef[k]is intentional as N[k] is amplified by these values beforehand and we want the basic noise estimate to calculate SNR). From this SNR can be calculated as:

, where S and N are, respectively, the amplitude of the input and noise.[[4]](#footnote-5)

Due to the SNR value changing very often, a low pass filter value is used by implementing the following: . is used for all future steps which involve SNR. The chosen k value was 0.95 but this can be change to liking.

A system similar to the noise M buffer is then set up with 2 buffers storing the maximum and minimum SNR:

|  |
| --- |
| SNR\_min[snr\_index] = **min**(SNR\_tm1,SNR\_min[snr\_index]); //snr\_index is what allows a circular buffer implementation  SNR\_max[snr\_index] = **max**(SNR\_tm1,SNR\_max[snr\_index]); |

These buffers are then switched every 2.5 seconds, with the oldest value being 10 seconds old.  
An across time min and max are then calculated to allow us to calculate a range:

|  |
| --- |
| snr\_inter\_min = **min**(**min**(SNR\_min[**0**],SNR\_min[**1**]),**min**(SNR\_min[**2**],SNR\_min[**3**]));  snr\_inter\_max = **max**(**max**(SNR\_max[**0**],SNR\_max[**1**]),**max**(SNR\_max[**2**],SNR\_max[**3**]));  snr\_inter\_range = snr\_inter\_max - snr\_inter\_min; //gives an idea of how the SNR range for the signal currently is |

The range is extremely useful. Firstly if it is found to be less than 3dB the VAD is disabled as the range is determined to be too small to accurately determine whether speech exists or not and we would risk removing speech as well as noise.

Secondly the range is used to calculate the cut-off SNR:

|  |
| --- |
| SNR\_Threshold = snr\_inter\_min+**0.2**\*snr\_inter\_range; |

In this case the SNR threshold would then be set to the lower 20%. The current implementation is a bit more complex and is similar to this SNR\_Threshold = snr\_inter\_min+Threshold\_coef\*snr\_inter\_range, where the value Threshold\_coef changes between 15% and 20% according to the SNR range.

Finally the last step is to activate the amplitude decrease when the input is detected to be under the calculated threshold:

|  |
| --- |
| **if** (VAD\_on&(SNR\_tm1 <= SNR\_Threshold)){  **for** (k=0;k<FFTLEN/2;k++){  C[k]= rmul(VAD\_coef,C[k]);//VAD\_coef is a value between 0 and 0.5 which reduces signal amplitude  }  } |

A step by step of this process can be found in the appendix under **Code Snippet 2**.

## Results



With VAD

With no VAD – only usual noise removal

Figure 3 Spectrogram of the file “factory2.wav” with and without VAD.

Figure 3 shows factory 2 being processed using the proposed VAD algorithm. As can be seen at silent periods the noise is effectively completely removed (simply being black indicating no sound presence). Applying this other sound samples also worked effectively as seen from **Table 1**.

|  |  |  |  |
| --- | --- | --- | --- |
| Sound file | Observation | Sound file | Observation |
| car1.wav | VAD sometimes deactivated due to low range | lynx2.wav | Worked well |
| factory1.wav | Worked well | phantom1.wav | Worked well |
| factory2.wav | Worked well – sometimes let a bit of noise through | phantom2.wav | Worked well – sometimes deactivated itself |
| lynx1.wav | Worked well | phantom4.wav | Worked to an extent – can get a bit confused and clip speech |

Table 1 shows how the VAD worked across all provided sound samples.

Implementing an adaptation algorithm which set the threshold % based on the range and average SNR values also helped the VAD lock in on more appropriate thresholds according to the input.

It was found that without correct adaptation algorithms some speech could be clipped, an effect commonly observed in SNR based voice activity detectors in low SNR conditions [4] (thus the motivation to disable the VAD when the SNR or SNR range is too low). However this VAD implementation can still prove useful in better estimating noise by only taking samples during periods in which speech is not detected.

# Future Improvements

Possible improvements would tackle the central issue of noise estimation which is the most crucial part for effective noise removal. One method would be to improve the smoothing function of the noise estimates (which enhancement 1 and 2 are concerned with) by having the value be dynamic. This is because currently the smoothing function is always biased for lower noise values and when noise amplitude increasing the estimates lag behind. Additionally periods of speech have their frequency bins be widened up (when viewed as a time function) leading to unwanted noise estimates. Thus one improvement would to have the time constant (which determines from ) decrease during periods of speech or very quick increases (depending on the implementation and the type of accuracy wanted) to “forget” faster about past values which may simply be anomalies.   


 [[5]](#footnote-6)

# Works Cited

|  |  |
| --- | --- |
| [1] | P. Čisar and S. M. Čisar, “Optimization Methods of EWMA Statistics,” 2011. [Online]. Available: http://www.uni-obuda.hu/journal/Cisar\_Cisar\_31.pdf. [Accessed March 2013]. |
| [2] | “Human speech fundamental frequencies,” in *Clinical Measurement of Speech and Voice*, London, Taylor and Francis Ltd., 1987, pp. 177,188. |
| [3] | J. Rodman, “The Effect of Bandwidth on Speech Intelligibility,” Polycom, 2006. [Online]. Available: http://docs.polycom.com/global/documents/whitepapers/effect\_of\_bandwidth\_on\_speech\_intelligibility\_2.pdf. [Accessed 2013]. |
| [4] | L. Ding, A. Radwan, M. S. El-Hennawey and R. A. Goubran, “Measurement of the Effects of Temporal Clipping on Speech Quality,” *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT,* vol. 55, no. 4, pp. 1197-1203, August 2006. |

# Appendix

## Code snippets

|  |
| --- |
| clc;  a = ones(256,1);  %choose what factor to set the values to  %and where this should start  lp\_ampli = 1.5;  lp\_freq = 10;  hp\_ampli = 1.2;  hp\_freq = 30;  for i=0:(lp\_freq-1)  a(i+1) = (lp\_freq\*lp\_ampli-(lp\_ampli-1)\*i)/lp\_freq;  a(256-i) = (lp\_freq\*lp\_ampli-(lp\_ampli-1)\*i)/lp\_freq;  end  a\_eq = (hp\_ampli-1)/(128-hp\_freq);  b\_eq = 1 - a\_eq\*hp\_freq;  for i=hp\_freq:128  a(i+1) = a\_eq\*i+b\_eq;  a(257-i) = a\_eq\*i+b\_eq;  end  plot(1:256,a)  axis tight  title('Alpha exaggeration coefficients')  xlabel('Index')  ylabel('Amplification factor')  formatSpec = '%1.16f,'; %set format to high precision to avoid rounding issues  fid=fopen('project\_pt1\RTDSP\lpfcoef.txt','w'); %open file  % define order of filter+1 here  % simpler and more intuitive way than    %define a and b coefficient arrays  fprintf(fid,'float lpfcoef[] = {');  fprintf(fid,formatSpec,a(1:length(a)-1));  fprintf(fid,'%1.16f};\n',a(length(a)));  fclose(fid); %close file |

Code Snippet 1 Matlab code to generate alpha exaggeration coefficients

|  |
| --- |
| 1. Calculate Noise Power 2. Calculate Input Power 3. Set SNR = 10log(Input\_Power/Noise\_Power) 4. Apply low pass filter on SNR: SNR\_lpf = 0.1\*SNR+0.9\*SNR\_lpfprev 5. Create minimum and maximum buffers 6. From these buffers calculate the SNR range (max – min) 7. From this range determine whether or not it is sensible to activate voice activity detection (values under 3 are not recommended to have VAD on) 8. Calculate the threshold: SNR\_threshold = SNR\_min + 0.2\*SNR\_range a 20% range is an experimental value found to work but any other appropriate value could be used 9. If (SNR\_lpf\_current < SNR\_Threshold) then decrease output signal by a constant as the algorithm has decided that no speech is occurring. |

Code Snippet 2 Pseudo-code to create a simple VAD which removes noise when no speech is occuring

*You need to write a report explaining how your program works, the evaluations you performed and the reasons for you choice of parameters. You should submit a copy of your source code as an appendix to your report (which does not count in the page limit). Your report should give formulae that specify precisely what quantities are calculated in your program. It is unlikely that a single set of parameters will be optimum for all types of speech and noise; your report should make clear what compromises you make in choosing your final algorithm. You must also try to give explanations of your understanding of why the enhancements work (or do not work) better than the simple algorithm. See the mark sheet for marking details.*

*Please keep the main body of your report to no more than 12 pages. This is ample space in which to describe what you have done, if you write clearly and concisely. (Remember that research papers, which usually contain around 1 year’s worth of work are usually only 6 pages long)*

1. While the noise frequency bins distribution is not actually Gaussian this is the best way to think about this solution [↑](#footnote-ref-1)
2. This is a reasonable approximation as 95% of values in a Gaussian distribution lie within two standard deviations. [↑](#footnote-ref-3)
3. The occasional “clunks” are what we are referring to here. [↑](#footnote-ref-4)
4. Note that here S does not represent signal as such but rather signal and noise. [↑](#footnote-ref-5)
5. Note: the input files have been down sampled to 8000Hz sampling rate for better comparison purposes [↑](#footnote-ref-6)