RTDSP Project

Contents

Program Function

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

The program works by assigning the 4 M_i 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 optimisations. 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 optimisation 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 optimisations

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

Additional Optimisations

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

First optimisation

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 [1]. Due to speech harmonics being important for speech intelligibility [2], 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 optimisation 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).

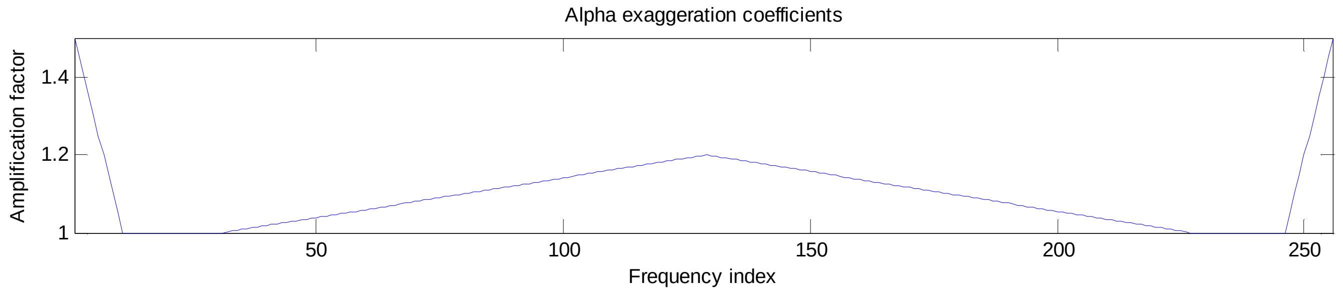

Figure 1 Visual plot of alpha coefficients use to exaggerate the noise at certain frequencies. The reason for the symmetric coefficients is due to the input of the FFT being symmetric. All coefficients are normalized thus can be multiplied with alpha to obtain the desired effect.

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 optimisation. 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.

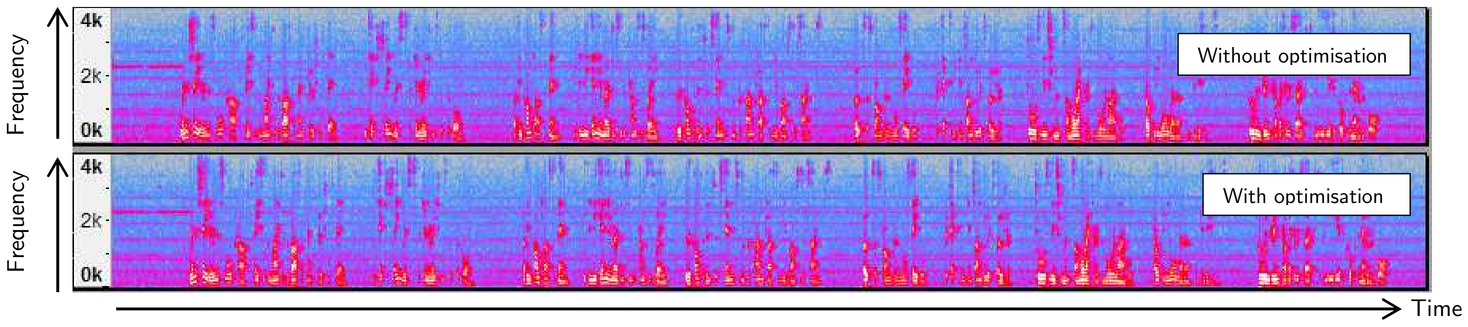

Figure 2 Spectrogram of the file lynx1.wav with and without the optimisation. The more intense the colour indicates a high presence of the frequency whereas a grey/blue colour implies the frequency components at the frequency are very low.

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 Optimisation (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[1]. 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 optimisation 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:

$SNR_{in} = 10\log_{10}\left(\frac{\sum_{i=0}^{FFTLEN/2} S_i^2}{\sum_{i=0}^{FFTLEN/2} N_i^2}\right)$, where S and N are, respectively, the amplitude of the input and noise.[2]

Due to the SNR value changing very often, a low pass filter value is used by implementing the following: $SNR_{lpf} = SNR_{calc} \times (1-k) + k \times SNR_{lpf\_prev}$. SNR_{lpf} 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:

[1] The occasional "clunks" are what we are referring to here.
[2] Note that here S does not represent signal as such but rather signal and noise.

```
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

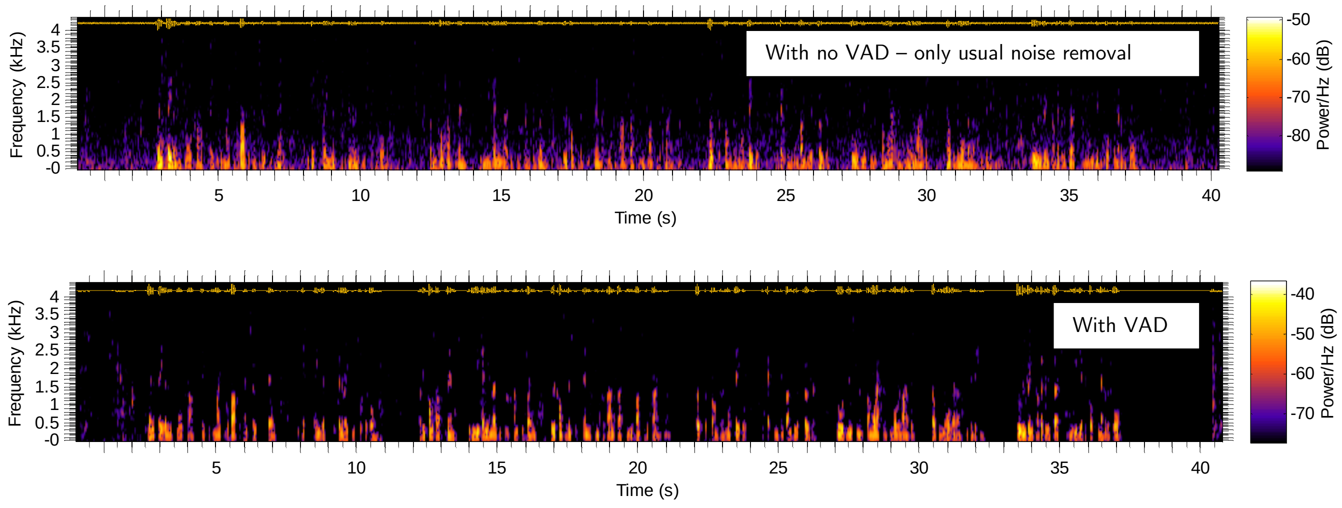

Figure 3 Spectrogram of the file "factory2.wav" with and without VAD.

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
lynx1.wav	Worked well	phantom4.wav	Worked to an extent – can get a bit confused

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.

Has the student shown significant understanding of the material beyond following the specific coursework instructions, or has the student done anything which deserves additional credit?
-Apx500 (does noise check)
-Matlab
-Audacity Spectgram

Works Cited

[1] "Human speech fundamental frequencies," in *Clinical Measurement of Speech and Voice*, London, Taylor and Francis Ltd., 1987, pp. 177,188.

[2] 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].

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)