Department of Electrical & Electronic Engineering
Imperial College of Science, Technology and Medicine

Project: Real-Time Speech Enhancement

Introduction

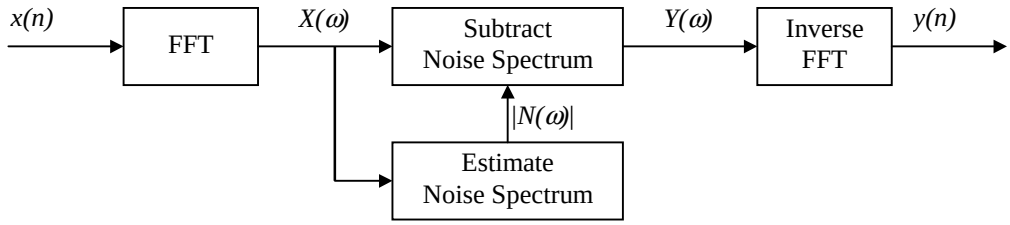
Telephones are increasingly being used in noisy environments such as cars, airports and undergraduate laboratories. The aim of this project is to implement a real-time system that will reduce the background noise in a speech signal while leaving the signal itself intact: this process is called *speech enhancement*.

Algorithm

Many different algorithms have been proposed for speech enhancement: the one that we will use is known as *spectral subtraction*. This technique operates in the frequency domain and makes the assumption that the spectrum of the input signal can be expressed as the sum of the speech spectrum and the noise spectrum.

The procedure is illustrated in the diagram below and contains two tricky parts:

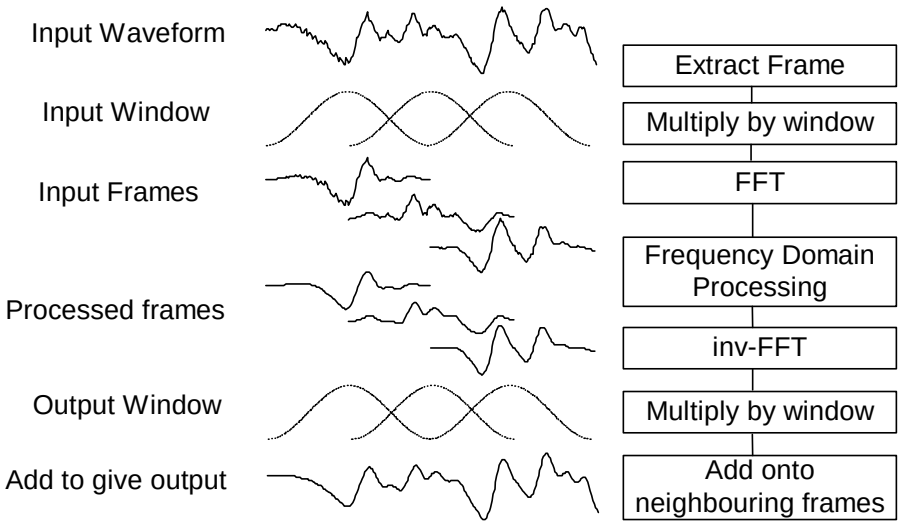
- estimating the spectrum of the background noise
- subtracting the noise spectrum from the speech



The sample rate of the system is 8 kHz and a 256-point Fourier transform is performed on the input signal every 64 samples (8 ms).

Overlap-Add Processing

To perform frequency-domain processing, it is necessary to split the continuous time-domain signal up into overlapping chunks called frames. After processing, the frames are then reassembled to create a continuous output signal. To avoid spectral artefacts, we multiply the frame by a window function before performing the FFT and again after performing the inverse-FFT.



The output signal is thus formed by adding together a continuous stream of 256-sample frames each of which has been multiplied by both an input and an output window. If we choose the windows to be the square root of a Hamming window:

$$\sqrt{1-0.85185\cos\left((2k+1)\pi/N\right)} \quad \text{for} \quad k=0,\cdots,N-1$$

then the overlapped windows will sum to a constant and the output signal will be undistorted by the framing process.

In the diagram above, each frame starts half a frame later than the previous one giving an *oversampling ratio* of 2. This normally gives acceptable results but can introduce distortion if the processing alters the gain of a particular frequency bin abruptly between successive frames. It is therefore more common to use an oversampling ratio of 4 in which each frame starts only a quarter of a frame after the previous one. In this case, each output sample is the sum of contributions from four successive frames.

Subtracting the Noise Spectrum

The basic idea is just to subtract the noise off the input signal:

$$Y(\omega) = X(\omega) - N(\omega)$$

Unfortunately we don't know the correct phase of the noise signal so we subtract the magnitudes and leave the phase of X alone:

$$Y(\omega) = X(\omega)\times\frac{|X(\omega)|-|N(\omega)|}{|X(\omega)|} = X(\omega)\times\left(1-\frac{|N(\omega)|}{|X(\omega)|}\right) = X(\omega)\times g(\omega)$$

We can regard $g(\omega)$ as a frequency-dependent gain factor, so this is really just a form of zero-phase filtering.

A further problem is that it is quite possible for the multiplicative factor in the above expression to go negative from time to time. To avoid this, we actually use the following formula:

$$g(\omega) = \max\left(\lambda, 1-\frac{|N(\omega)|}{|X(\omega)|}\right)$$

where the constant λ is typically 0.01 to 0.1.

Estimating the noise spectrum

When someone is speaking, they inevitably have to pause for breath from time to time. We can use these gaps in the speech to estimate the background noise. One way of doing so is to design a Voice Activity Detector (VAD) which identifies whether or not speech is present in a signal. Unfortunately a reliable VAD is exceptionally difficult to make and so we choose an easier approach [4].

For each frequency bin in the Fourier transform, we determine the minimum magnitude that has been present in any frame over the past ten seconds or so. Under the assumption than no one ever talks for more than ten seconds without a break, this spectral minimum will correspond to the minimum noise amplitude that occurred during non-speech intervals. Since this will underestimate the *average* noise magnitude, we must multiply by a compensating factor before using it in the formulae above.

Determining the precise minimum over the past ten seconds requires storing all spectra from this interval which is unrealistic. We can determine an approximate minimum with far less storage as follows. We store four estimates of the minimum spectrum $M_i(\omega)$ where i=1,2,3,4. For each frame, we update M_1 by:

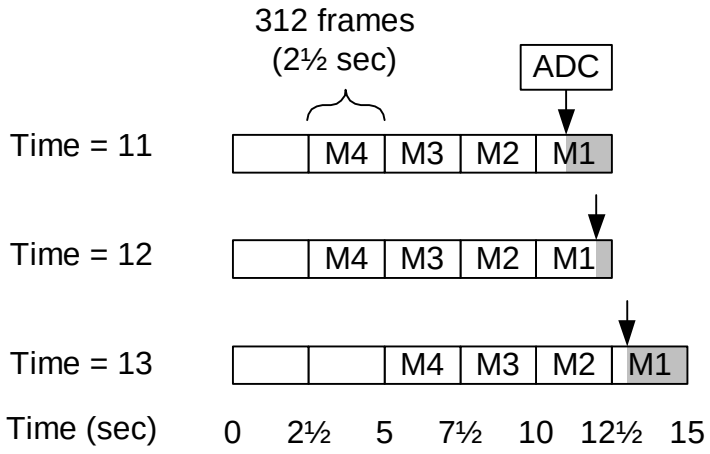
$$M_1(\omega) = \min\left(\left|X(\omega)\right|, M_1(\omega)\right)$$

Every 2.5 seconds, we transfer $M_i(\omega)$ to $M_{i+1}(\omega)$ and set $M_1(\omega)=|X(\omega)|$.

We can therefore estimate the noise spectrum as

$$\left|N(\omega)\right| = \alpha \min_{i=1..4}\left(M_i(\omega)\right)$$

where the factor α corrects for the underestimation of the noise discussed above. The factor α may need to be as high as 20 but the use of enhancement (1) described at the end of this sheet will allow it to be reduced to around 2 and will give more reliable estimates.



The diagram above indicates the situation at 11 sec, 12 sec and 13 sec after the start of the program. In the first case, at time 11 seconds, M4, M3 and M2 contain the minimum spectrum that occurred in the three intervals 2½ to 5 seconds, 5 to 7½ seconds and 7½ to 10 seconds (the minimum of 312 frames in each case). M1 contains the minimum spectrum that occurred between 10 and 11 seconds. Taking the minimum of all four buffers therefore gives the minimum spectrum over the 8½ second interval from 2½ to 11 seconds.

In the second case, M4, M3 and M2 are unchanged, but M1 now contains the minimum from 10 to 12 and so taking the minimum of all four buffers gives the minimum spectrum over the 9½ second interval from 2½ to 12 seconds.

At time 12½, we transfer M3 to M4, M2 to M3 and M1 to M2, so in the third case the minimum of all four buffers gives the minimum spectrum over the 8 second interval from 5 to 13 seconds.

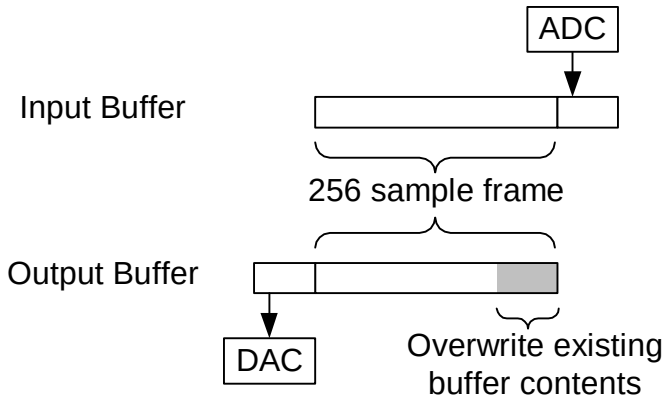
Skeleton Program

A skeleton program is available as **enhance0.c**. This program performs the input/output buffering and interrupt handling that are needed. It does not perform the FFTs, noise estimation or spectral subtraction: that is for you to implement. All the constant parameters associated with the algorithm are specified in a definitions block near the start of the program: it is bad practice to bury peculiar constants in the middle of your code so you should follow this procedure for any other constants that you need to add. Some of the constants are used to define buffer lengths while others are used in arithmetic expressions. Some of the latter (e.g. INGAIN and OUTGAIN) are copied into global variables in the initialisation section of the main program: the reason for this is so that they can be altered in real time while the program is running. Note that when defining a constant to equal an expression, you should enclose the expression in parentheses to avoid unexpected results if it is used in an arithmetic expression.

To get the program going, you should create a new directory and copy enhance0.c into it. Then create a new project (**Project→New**) called enhance.mak. Next create a configuration file (**File→New→DSP/BIOS Configuration→dsk6711.cdb**) and map the codec interrupt (**Expand HWI manager, right-click HWI_INT9, select properties, set: function=_XINT0_HWI and "use dispatcher"=yes**) and save the file as enhance.cdb. You need to change the output filename so that it matches that of the configuration file (**Project→Options→Linker, set output filename = enhance.out**). Finally add the following files to the project: enhance.cdb, enhancecfg.cmd, enhance0.c (**Project→Add file**).

Input/Output Buffers

With an oversampling rate of 4, we need to process six quarter-frames at any given moment:



While the A/D converter is transferring samples into the current ¼-frame, we process the previous four ¼-frames as indicated in the diagram. We add the processed frame onto the existing contents of the output buffer (from previous frames). For the last quarter of the frame, there is no data from previous frames and so we overwrite whatever information happens to be in the buffer instead of adding onto it. While all of this is happening, the previous ¼-frame, which is now completely specified, is sent to the D/A converter. Our algorithm therefore has a 1¼-frame delay that is independent of the processor speed (this is in addition to the codec delay which is quite large).

Buffer Synchronisation

Notice that both the input and output buffer need only be five ¼-frames long. We implement them as circular buffers and reset the variable **io_ptr** to zero in the interrupt service routine whenever it increments beyond the end of the buffer. When we have finished processing a frame, we need to wait until the ADC and DAC have finished the current ¼-frame. We do this by checking the value of **io_ptr/64** to see which section of the buffer it has reached. We also use **io_ptr** to calculate the value of **cpufrac** which tells us what fraction of the available CPU time we are using: you can monitor this from a watch window.

Objectives and Milestones

The aim of this project is to implement the spectral subtraction technique described above and then to improve it to obtain the best possible performance on the test files provided.

You may find it easier to implement the two parts of the algorithm independently at first and check that they work correctly. As an initial test, you could implement a simple frequency-domain filter by setting some of the FFT spectrum values to zero.

The files that you need are available from any Departmental PC by mapping the network path **\\reserver\sp_data** onto a convenient drive. Many of them have also been copied onto the PCs in the lab: you will need to substitute the appropriate value for ***** in the paths specified below.

Test Data

A number of test signals are available in ***\C6x\enhance\data*.wav** with descriptions in ***\C6x\enhance\signals.txt**. You can use Windows to replay the test signals via the soundcard.

Enhancements

You may wish to evaluate some of the following enhancements that researchers have suggested. In my experience some of them are a good idea but others make things worse.

1. Use a low-pass filtered version of $|X(\omega)|$ when calculating the noise estimate. Note that this low-pass filter is operating on successive frames not on the speech samples themselves. If $P_t(\omega)$ is the low-pass filtered input estimate for frame *t*, then $P_t(\omega) = (1-k)\times|X(\omega)| + k \times P_{t-1}(\omega)$ where $k = \exp(-T/\tau)$ is the *z*-plane pole for time constant τ and frame rate T=16 ms. If you do this, you will be able to reduce the oversubtraction factor α described above. The value of T is defined in the C program as **TFRAME**: you can calculate the value of *k* in the initialisation section of the main program. A plausible time constant to use is in the range 20 to 80 ms.
2. The above low-pass filtering can be performed in the power domain rather than the magnitude domain. That is, you can low-pass filter $|X(\omega)|^2$ and then take the square root to find $P(\omega)$.
3. Low pass filter the noise estimate $|N(\omega)|$ to avoid abrupt discontinuities when the minimization buffer length changes. This will only have a noticeable effect if the noise level is very variable.
4. Instead of setting $g(\omega) = \max\left(\lambda, 1-\frac{|N(\omega)|}{|X(\omega)|}\right)$, set it to $\max\left(\lambda\frac{|N(\omega)|}{|X(\omega)|}, 1-\frac{|N(\omega)|}{|X(\omega)|}\right)$, $\max\left(\lambda\frac{|P(\omega)|}{|X(\omega)|}, 1-\frac{|N(\omega)|}{|X(\omega)|}\right)$, $\max\left(\lambda\frac{|N(\omega)|}{|P(\omega)|}, 1-\frac{|N(\omega)|}{|P(\omega)|}\right)$, or $\max\left(\lambda, 1-\frac{|N(\omega)|}{|P(\omega)|}\right)$. In all cases you may wish to calculate $P(\omega)$ using a different (probably longer) time constant than used for enhancement (1).
5. Calculate g in the power domain rather than the magnitude domain, i.e. set $g(\omega) = \max\left(\lambda, \sqrt{1-\frac{|N(\omega)|^2}{|X(\omega)|^2}}\right)$ or a similar modification of an expression from (4).
6. Deliberately overestimate the noise level (by increasing α) at those low frequency bins that have a poor signal-to-noise ratio. This *oversubtraction* technique can reduce the "musical noise" artifacts that are introduced by spectral subtraction.
7. An alternative way of estimating $N(\omega)$ that requires much less storage space is to low-pass filter the input (as in enhancement 1 above) and then calculate:

$$P(\omega) = \frac{|X(\omega)| - \beta |X_{t-1}(\omega)|}{1-\beta}$$

$$|N(\omega)| = \begin{cases} (1-\gamma)P(\omega) + \gamma |N_{t-1}(\omega)| & \text{for} \quad |X(\omega)| > |N_{t-1}(\omega)| \\ |X(\omega)| & \text{for} \quad |X(\omega)| < |N_{t-1}(\omega)| \end{cases}$$

where the subscript *t*–1 denotes the value in the previous frame.

Typical values of β and γ correspond to time constants of around 0.2 and 4 seconds respectively.

8. Evaluate different frame lengths: according to [1], short frames sound rough with increased musical noise, while long frames sound slurred.

9. Reduce musical noise by applying the "residual noise reduction" described in [2]: if $\frac{|N(\omega)|}{|X(\omega)|}$ exceeds some threshold, then replace $Y(\omega)$ by its minimum calculated value in three adjacent frames (this entails adding an additional 1-frame delay since you need $Y(\omega)$ from the next frame in order to calculate the output in the current frame).

10. Several other possible enhancements are described in the references listed below.

Assessment

Towards the end of the project period, we will evaluate your program by listening to its effect on a number of test files. You are not allowed to have different programs or parameter values for different test files.

You need to write a report (one report per group) explaining how your program works, the evaluations you performed and the reasons for you choice of parameters. You will also need to submit a copy of your source code.

References

These references are available in directory ***\C6x\enhance\refs**.

[1] Berouti,M. Schwartz,R. & Makhoul,J.,"*Enhancement of Speech Corrupted by Acoustic Noise*", Proc ICASSP, pp208-211, 1979.

[2] Boll,S.F.,"*Suppression of Acoustic Noise in Speech using Spectral Subtraction*", IEEE Trans ASSP 27(2):113-120, April 1979.

[3] Lockwoord, P. & Boudy,J.,"*Experiments with a Nonlinear Spectral Subtractor (NSS), Hidden Markov Models and the projection, for robust speech recognition in cars*", Speech Communication, 11, pp215-228, Elsevier 1992

[4] Martin, R., "*Spectral Subtraction Based on Minimum Statistics*", Signal Processing VII: Theories and Applications, pp1182-1185, Holt, M., Cowan, C., Grant, P. and Sandham, W. (Eds.), 1994

Mike Brookes, May 2001