DCSP: Implementing Energy Endpoint Detection

Kieran Molloy

January 2020

Abstract

Using energy endpoint detection to eliminate noise and silence from a speech signal by quantising the signal and cutting the section of speech into a new file

1 Introduction

The supplied sound file is a .wav file, and is represented in Figure.1 this is a simple filestream data format where the data is simply stored as numerical values representing the amplitude, it is often considered the simplest audio format and as such is very useful for the task at hand. A .wav file can be read and written with ease, as it was partially designed for this purpose by IBM and Microsoft *WAV standard* 2019.

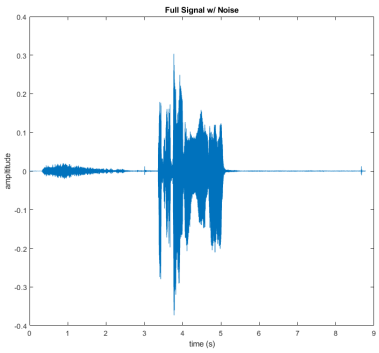

Figure 1: Original Signal

When the file is read, it will be converted into numbers and then it will be ready for processing. When processing has finished, it can simply be written to a new file and stored. The processing that will be perfomed is energy endpoint detection.

2 Method

The file that is being worked is 8 seconds long and consists of a period of noise, followed by a pause then a woman saying "Let us go then, you and I" followed by some more noise. A standard threshold endpoint detection algorithm will be fooled by the first section of noise, and will potentially cut the speech section short, even when using a threshold value. Therefore the method that this paper will use is the energy endpoint detection algorithm, which uses average signal characteristics over short time frames. Mingham 2019

2.1 Energy Endpoint Algorithm

Performing the energy endpoint algorithm requires three stages, quantisation, forwards pass and backwards pass.

The quantisation stage goes through the signal and calculates frame energies, Algorithm 1 details how frame energy is calculated, which gets the signal ready for the energy endpoint detection.

Algorithm 1 Frame Energy Calculation

1: **for** $i \leftarrow 1, nf$ **do**
2: $\quad f \leftarrow 0$
3: $\quad$ **for** $j \leftarrow 1 + (i-1)nspf, i \times nspf$ **do**
4: $\quad\quad f \leftarrow f + s_j^2$
5: $\quad E_i \leftarrow f$

The general idea of the function, is based on the mathematical principal of x^2, if $x < 0$ then $x^2 \to 0$, whereas if $x > 0$ then $x^2 \to \infty$. This causes speech, or higher amplitudes, to have massive energies compares to noise, or lower amplitudes, which make speech easily detected due to their energy values.

The second stage is a forwards pass, this stage searches for the start of speech by iterating through finding a value higher than a predefined threshold value z. Ingle 2017

1. Find the maximum value of all frame energies, call it E_{max}.
2. Set a threshold value, Z_f, for noise is set by comparing $E - 1$ and E_{max}. A sensible expression for Z_f is,
$$Z_f = E_1 + c(E_{max} - E_1) \tag{1}$$
where c lies between 0 and 1. Take $c = 0.1$ (which sets the threshold just above the energy of the first frame).
3. Compare the energy in each frame in to Z_f. The first frame where $E_j > Z_f$ gives the start of the word at (approximately) time $t = (j-1) \times w$
4. The end of a word is detected similarly (a neat way of using the same code is to reverse the signal).

Algorithm 2 Forwards Search

```
e_0 ← E[1]
e_max ← max(E)
c ← 0.1
Z ← e_0 + c(e_max − e_0)
for i ← 1, nf do
    if E_i > z then
        start ← 1 + (i − 1)nspf
```

The third stage is a backwards pass, this stage reverses the signal (or as Algorithm 3 does, iterates backwards) searching for a value higher than a predefined threshold value. This will become the ending of speech. As the explanation shows, the forwards and backwards passes are very similar algorithms, that perform effectively the same task.

1. Using the previous e_0, E_{max} and c values. Set a new threshold value, Z_b. The expression that will be used for Z_b is,

$$Z_b = E_{nf} + c(E_{max} - E_0) \tag{2}$$

2. Compare the energy in each frame in to Z_b. The first frame where $E_j > Z_b$ gives the start of the word at (approximately) time $t = (j-1) \times w$ reverse the signal).

Algorithm 3 Backwards Search

```
Z ← e_nf + c(e_max − e_0)
for i ← nf : −1 : 1 do
    if E_i > z then
        end ← 1 + i × nspf
```

The final solution should also track indexes of significant points to perform other operations such as plotting or other signal processing methods.

3 Results

Implementing Algorithms 1,2 & 3 from Section 2.1 in MATLAB r2018a. A full copy of the program is available in Appendix A

The signal energy levels are represented in Figure.2, where until 170 seconds (3.38 seconds/ 149059th frame) in the energy level is ≈ 0 and negligible, and the same is seen after 255 seconds (5.02 seconds/221383rd frame) with strong peaks in the middle. This shows that the energy calculation algorithm is working as designed, it is minimising the noise and maximising the speech. Using the forwards pass algorithm, defined in 2.1, the frame where speech commences is recorded as frame 149059. The backwards pass algorithm finds the frame where speech terminates as frame 221383. This gives a window of speech that can then be cut out, since the frame id's are already known it is simple matrix range selecting. The output of this is in Figure.3 and shows only the speech section of the original signal. Starting at 3.38

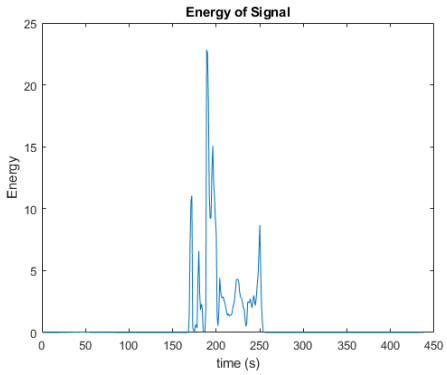

Figure 2: Energy Levels

seconds and ending at 5.02 seconds

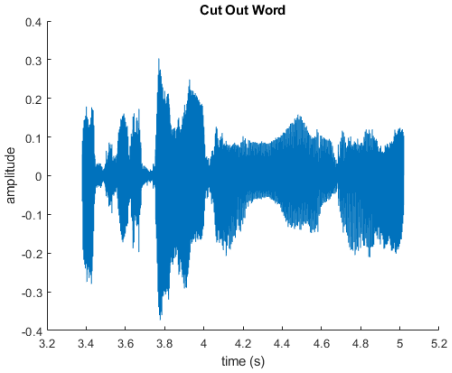

Figure 3: Speech Cutout from Original Signal

4 Conclusions

The energy endpoint algorithm can seem complicated at first but when broken down is very simple, the algorithm is able to find the start and endpoint of speech and therefore eliminate audible noise. The next step would be to introduce Crossing Rate analysis in conjunction with the Energy Endpoint to see if it is able to attain more accurate start and endpoints, however Crossing Rate Analysis brings its own problems where it could think some speech is noise, or any noise that does not have a high frequency could be mistaken for speech. The same idea could be implemented by tweaking with the z_f and z_b threshold values, the sweet spot is the lowest possible with all noise still detected and removed.

References

Ingle, John G. Proakis; Vinay K. (2017). *Digital signal processing using MATLAB: a problem solving companion*. Cengage Learning.

Mingham, Clive (2019). *Lecture Notes and Exercises for DIGITAL COMMUNICATIONS & SOUND PROCESSING*. MMU.

WAV standard (2019). Sustainability of Digital Formats: Planning for Library of Congress Collections. URL: `https://www.loc.gov/preservation/digital/formats/fdd/fdd000001.shtml` (visited on 07/11/2019).

A Matlab Code

```
% Kieran Molloy Submission for MMU Year 3: Digital Communications & Sound
% Processing Coursework Part 1

% DISCLAIMER
% Adaption of Clive Mingham's Energy Endpoint

% Outline of Solution;
% INPUT: .wav file
%   Quantise
%   Perform Energy Endpoint Detection
%   Cut Signal according to Endpoints
% OUTPUT: .wav file
%          Graphs

clear; clc; clf
% Read File
readFilename='SP1_2019_20.wav';
[s,fs]=audioread(readFilename);

% Number of Samples
N=length(s);
% Sampling Interval
T=1/fs;
% Length of Signal
P=N*T;
% T vector
t=0:T:P-T;

% Initial Plot
figure(1)
subplot(2,2,1)
plot(t,s)
ylabel('ampltitude')
xlabel('time (s)')
title('Full Signal w/ Noise')

% Frame Width
w=0.02;
% Number of Samples per Frame (NSPF)
Nspf=floor(w/T);
% Number of Frames (NF)
Nf=floor(P/w);
% Element-wise square function
```

```
s2=s.*s;
E=zeros(1,Nf); % better for memory
%% Energy Calculations
% Energy loop
for i=1:Nf
  summ=0;
  for j=1+(i-1)*Nspf:i*Nspf
    s2=s(j).*s(j);  %squaring s's
    summ=summ+s2;  %summing
  end
  E(i)=summ;  %energy for each frame
end

% nice plot stuff
frames=zeros(1,Nf);  % better for memory
for i=1:Nf
   frames(i)=i;
end
subplot(2,2,2)
plot(frames,E);
ylabel('Energy')
xlabel('time (s)')
title('Energy of Signal')

% Initial Energy
e0=E(1);
% Maximum Energy
maxE=max(E);
% Threshold Value
c=0.1;
% Threshold Formula
Z=e0+c*(maxE-e0) ;

% Initialise Search Variables
Istart=0;
Iend=N;

% FORWARDS
for i=1:Nf
    if (E(i)>Z)

        Istart=1+(i-1)*Nspf;
        break
    end
```

```
end

% BACKWARDS
% Re-Define Z for E(Nf)
Z=E(Nf)+c*(maxE-e0);

for i=Nf:-1:1
    if (E(i)>Z)
        Iend=1+i*Nspf;
        break
    end
end

% Actual Time Values for Start & End
tstart=(Istart-1)*T;
tend=(Iend-1)*T;
% Cut Signal & Time Array
scut=s(Istart:Iend);
tcut=t(Istart:Iend);

% Plot
subplot(2,2,3:4)
hold on
plot(tcut,scut)
hold off
title('Cut Out Word')
ylabel('amplitude')
xlabel('time (s)')

% Write to file
writeFilename = 'cutsignal.wav';
audiowrite(writeFilename,scut,fs);
```