DSP LAB 10

104061171　紀伯翰

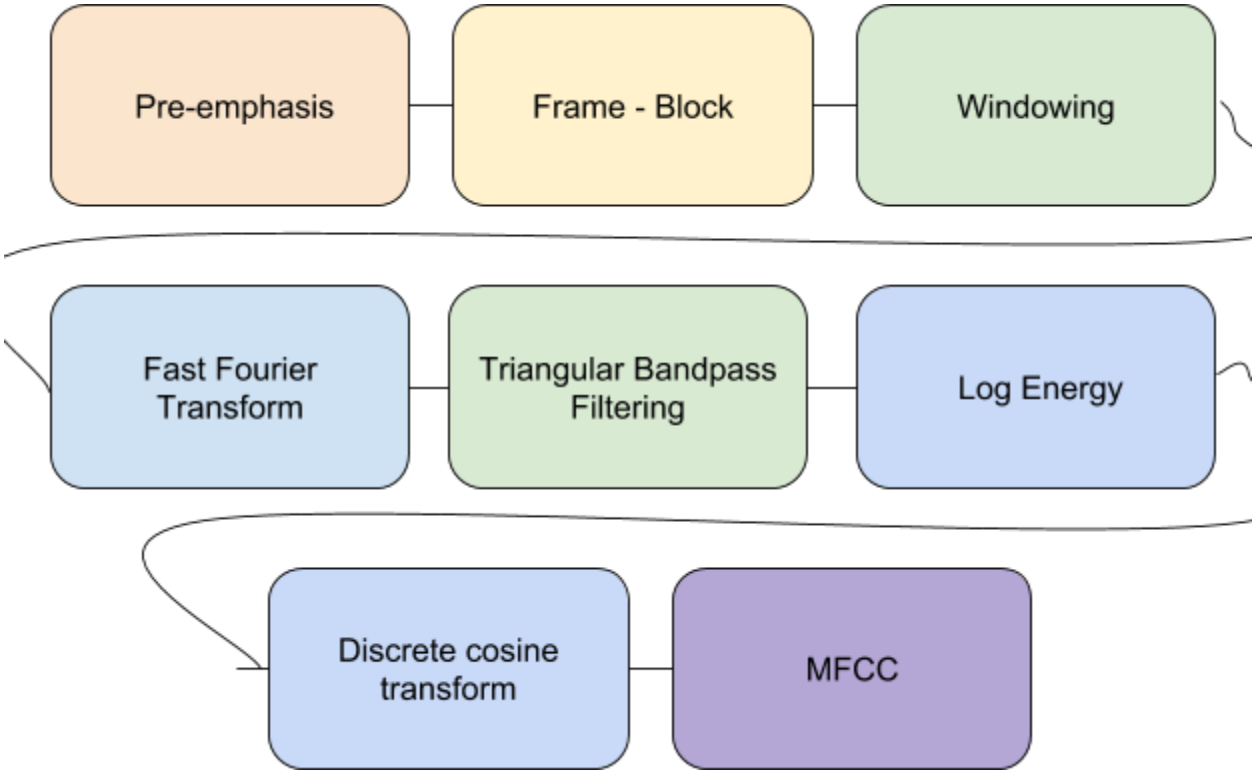
1.Abstract

MFCC(Mel-Frequency Cepstrum Coefficient)是一種提取聲音特徵的方法，普通訊號的domain通常是轉換到frequency domain繼續來做訊號處理，而MFCC使用的domain是近似於人耳系統，稱為cepstrum，為何要使用不同的domain呢，對於人耳來說，頻域上有很多無法辨識的訊號頻段，所以信號透過20個以上的filters，轉換到梅爾刻度上，之後再轉換到cepstrum上，可以將對於人耳可以聽出來的頻段做出更有效的訊號處理，所以使用mfcc處理，可以更有效的處理聲音訊號，並且將聲音的特徵提取出來。

2.Goal of this Lab

本次實驗，要實作mfcc訊號處理，在mfcc內有許多block diagram，幾個比較重要的部份，梅爾刻度與頻率的轉換，以及梅爾三角刻度的濾波器，還有透過 discrete cosine transform 並將圖畫出來觀察其特徵。

3.Method



Pre-emphasis

Filter　H(z)=1-a*z-1　<---> s2(n) = s(n) - a*s(n-1)

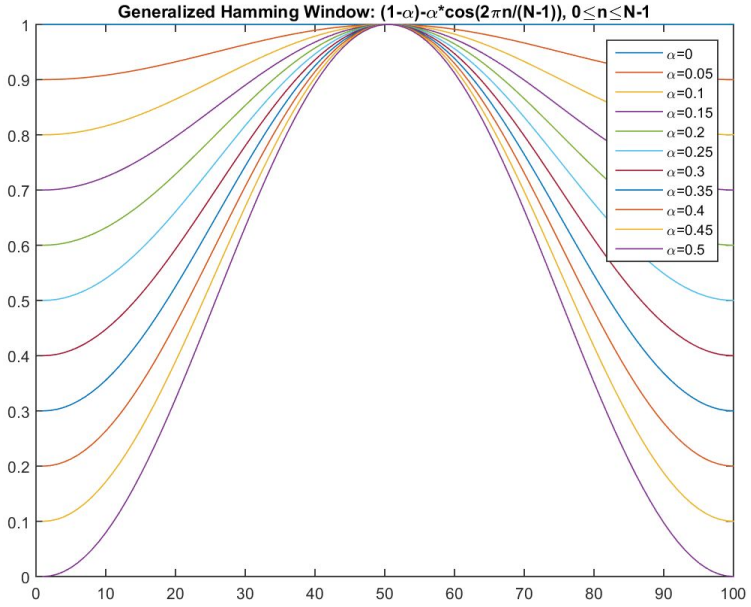
透過這filter，相較於通過一個高通濾波器，使人耳可以聽到高頻的訊號變得更清楚(濾掉低頻）。

Frame Block

將音訊切成一個一個的音框，又稱frame，以frame step 間隔取frame，通常frame step會是frame 長度的1/3 到 frame 長度的1/2，為的是避免音框內的訊號變化過多。所以相鄰音框之間會有重複的區域。

Windowing

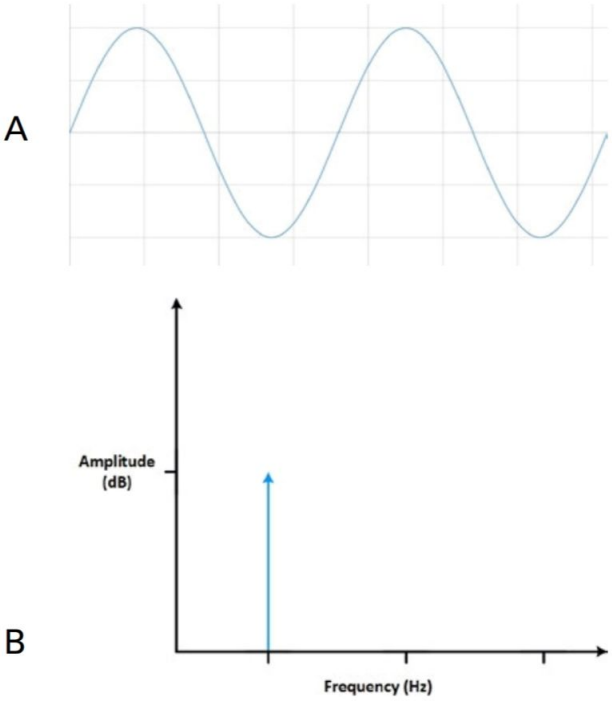
將這些音框乘上hamming window，為的是增加音框左邊以及右邊的連續性，為何會這樣說呢？來看看hamming window的數學式子 W(n, a) = (1 - a) - a cos(2pn/(N-1))，0≦n≦N-1，恩，我也看不出什麼毛，那我們從圖形來看



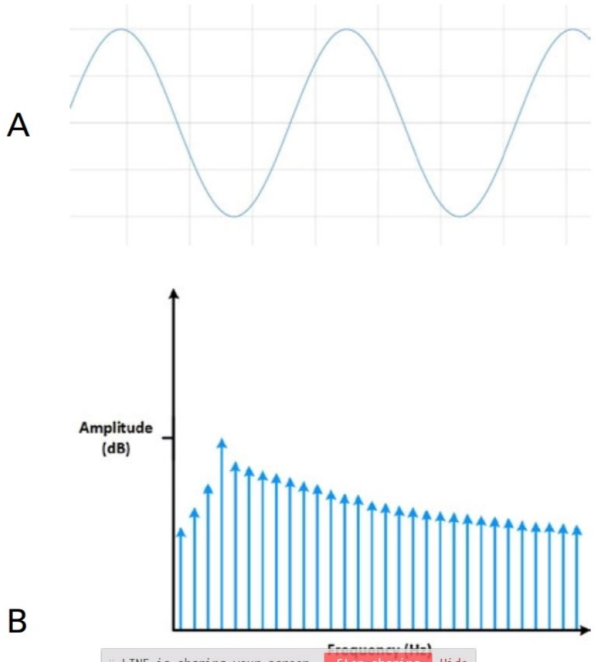
從圖形中，我們可以看到當我們每一個音框經過hamming window的時候，左右兩端的訊號值會變小，甚至趨近於0，那這事情的功用是什麼呢？因為後端我們會對每一個音框做快速傅立葉轉換，那從文獻中得知，如果finite訊號不是一個整數週期的訊號的話，進行fft會多出許多我們不要的雜訊，所以說，透過hamming window，原訊號經過之後，信框左右兩端的值會趨近於近於連續，並且減少spectral leakage的效應，因此我們可以有效衰減那些經過fft後不必要的雜訊影響再進行後續的訊號處理。

Example:

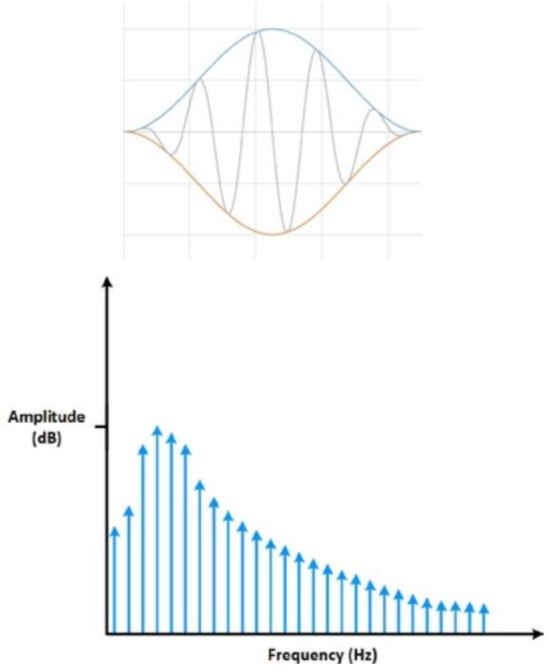
1.訊號左右兩端連續，對週期訊號取整數週期



2.訊號兩端沒有連續，對週期訊號取非整數週期



所以當將訊號套上 Hamming Window

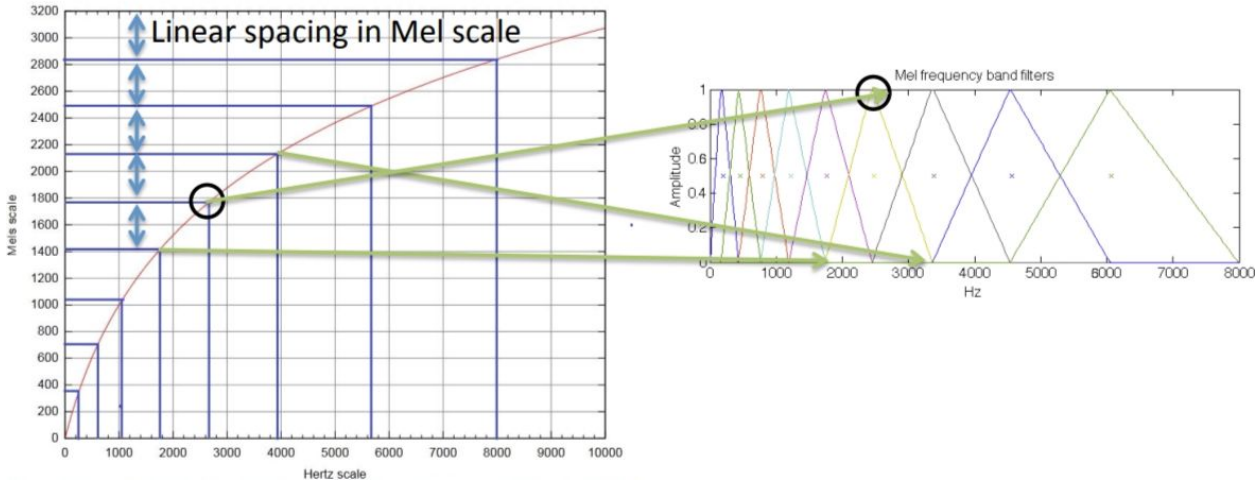


可以看到變成一個比較折衷的效果，spectral leakage的效應也會減少許多。

FFT
Fast Fourier Transform再取abs(頻譜)
將訊號做fft轉換後再取絕對值。

Triangular Bandpass Filtering

因為訊號的頻譜非常廣，不是所有頻段都是聲音適用的頻段，而且訊號用的頻譜刻度，也和人耳受器所使用的聲音刻度不太一樣，聲音的刻度並不是線性的，是可以用梅爾刻度來近似的，為了更好的模擬人耳接收聲音的狀況，我們將頻譜map到梅爾刻度，較近似人耳接收聲音的domain，除此之外，透過triangular filter，對頻譜進行平滑化，並消除諧波的作用，突顯原先語音的共振峰，因此一段語音的音調或音高，是不會呈現在 MFCC 參數內，換句話說，以 MFCC 為特徵的語音辨識系統，並不會受到輸入語音的音調不同而有所影響，更能夠有效率的提出聲音的feature.



而這些filter在mel刻度上也是線性平均分佈的。

log Energy

將原本取絕對值的訊號做平方，並且取log，取log更能夠貼切的近似人耳接收聲音的感受。

DCT(Discrete Cosine Transform)

將取完log的訊號做DCT，做這件事的目的是希望可以將能量訊號轉至一個近似time domain的domain來看，那我們稱它cepstrum，加上前面有使用mel-scale轉換，總稱Mel-scale Cepstrum，$C_m=S_{k=1}^{N}\cos[m*(k-0.5)*p/N]*E_k$, m=1,2, ..., L，那到這邊就完成了我們的mfcc，可以看到在這個domain下，我們可以將20個三角bandpass filters的feature，當成每個frame的attributes，（代表我們把一個frame使用20個三角bandpass filter的數值做表示），那每個frame內都有這些attribute，代表這個frame在不同mel刻度下的能量響應是多少，extract出這個frame內的features。

MFCC

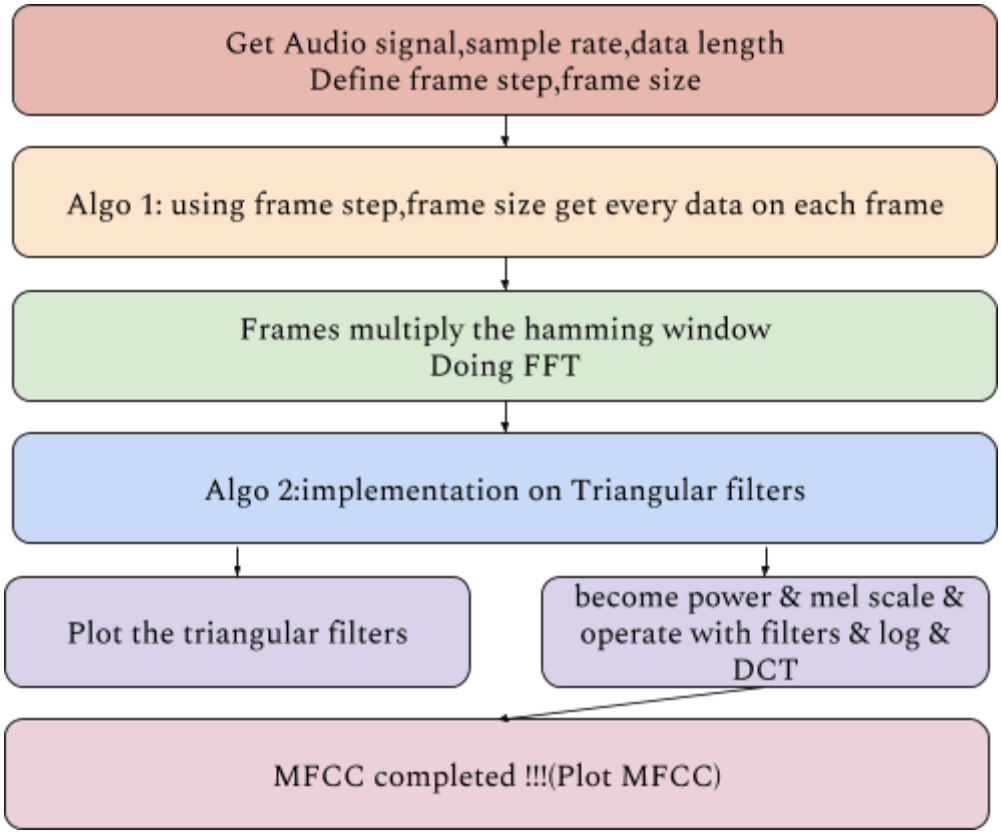
透過以上所有流程，完成MFCC !!!

4.Pseudo Code

```
func pre_emphasis(signal,coefficient=0.95):
   return signal[0]+(signal[1:end]-coefficient*signal[start:-1])

func mel2hz(mel):
   hz <- 700 * (10 **(mel / 2595 )-1)
   return  hz

func hz2mel(hz):
   mel <- 2595 * np.log10(1+hz /700)
   return  mel

func get_filter_banks(filters_num,NFFT,samplerate,low_freq=0,high_freq=None):
   set low_mel by call func hz2mel(low_freq)
   set high_mel by call func hz2mel(high_freq)
   set mel_points <- np.linspace(low_mel,high_mel,filters_num+2)
   set hz_points by call func mel2hz(mel_points)
   bin <- floor((NFFT+1)*hz_points/samplerate)
   Fbank <- get zero_array([filters_num,int(NFFT/2+1)])

   for i from 1 to  filters_num+1:
      f_minus <- int(bin[i-1])
      f_m <- int(bin[i])
      f_add <- int(bin[(i+1)])
      for j from f_minus to f_m:
         fbank[i-1,j] <- (j-bin[i-1])/(bin[i]-bin[i-1])
      for j from f_m to  f_add:
         fbank[i-1,j] <- (bin[i+1] - j)/(bin[i+1]-bin[i])

   return fbank

if __name__ == '__main__':

   signal,fs = sf.read('./SteveJobs.wav')

   fs=fs                                          #SampleRate
   set Signal_length to len(signal)               #Signal length
   set win_length to   0.025                      #Window_size(sec)
   set win_step to   0.01                         #Window_hop(sec)
   set frame_length to integer(win_length*fs)     #Frame length(samples)
   set frame_step to integer(win_step*fs)         #Step length(samples)
   set emphasis_coeff to 0.97                     #pre-emphasis para
```

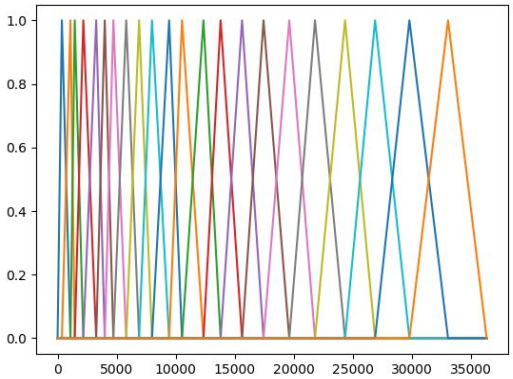
```
set filters_num to   22                          #Filter number

set NFFT to 256
set low_freq to 0
set high_freq to integer(fs/2)



signal <- pre_emphasis(signal)

frames_num <- 1+integer(ceil((1.0*signal_length-frame_length)/frame_step))

pad_length <- integer((frames_num-1)*frame_step+frame_length)
zeros <- np.zeros((pad_length-signal_length,))
pad_signal <- concatenate signal and zeros

get index on every frame by the initial value  frame step and frame length
frames <- fill in pad_signal by the index.
frames <- frames multiply hamming_window(frame_length)

complex_spectrum <- doing real Fast Fourier Transform on the frame
absolute_complex_spectrum <- absolute(complex_spectrum)
fb <- by call func get_filter_banks(filters_num,NFFT,fs,low_freq,high_freq)

xaxis <- get array from 0 to (length(fb[3])) * (1.0*fs/len(fb)))

for i from 0 to len(fb):
    plot(xaxis,fb[i])

powerframe <- ((1.0 / NFFT) multiply ((absolute_complex_spectrum)^2)

fb <- powerframe multiply transpose(fb)
if fb element equal 0
    assign small error
else
   keep original element
end
feat <- log10(fb)
Feat <- Discrete Cosine Transform(feat, norm='ortho')[:,:filters_num]

plot(feat)
Show picture()
```

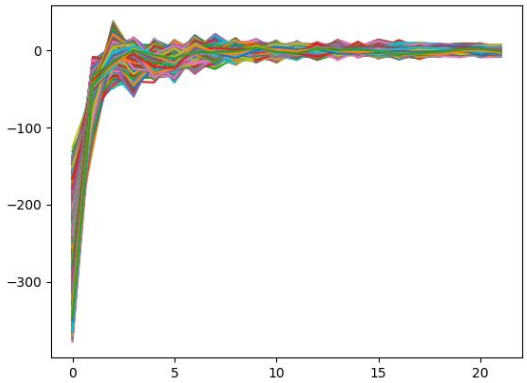
5.Flow Chart



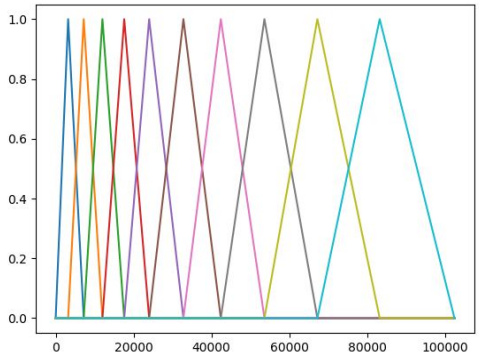
6. Result

triangular filters number: 22





triangluar filters:10



7.References

http://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf?fbclid=IwAR3nHnoaJi9dVFhrCfXGgoW5KDK0kpdB3QjbL693hFhqTkcJKm9-57VSpS0

http://mirlab.org/jang/books/audiosignalprocessing/speechFeatureMfcc_chinese.asp?title=12-2%20MFCC

8.Suggestion

None