FCU

ePaper

逢甲大學學生報告 ePaper

音訊信號的隨機性程度測試

Test for the degree of randomness of audio signals

作者：吳宜峰

系級：資訊四丙

學號：D0683410

開課老師：林育德

課程名稱：生醫信號處理

開課系所：自動控制工程學系生醫學程

開課學年：109 學年度 第二學期

摘要

本次實驗的目的是要分析音訊信號中不同片段的隨機性程度，並判斷哪些片段屬於隨機信號或是不屬於隨機信號，最後討論隨機信號跟語音中的清音(unvoiced sound)與濁音(voiced sound)特徵的關係，也會討論取出片段的長度對於隨機信號判斷的關係。

本次實驗是根據課本第 116 頁的方法，先標記音訊信號中所有的轉折點，再分片段統計片段中轉折點的個數，最後利用音訊信號中局部片段的轉折點個數佔此片段的比例來判斷此片段的信號是否為隨機信號。

本次實驗使用的測試信號為兩段錄音，音訊內容分別為"safety"與"吳宜峰"，錄音的工具是手機(OPPO R11s)，音訊採樣頻率為 44100 Hz。

觀察結果後可以發現特徵跟理論大致有符合。聲音如果是 unvoiced sound 與 plosive sound 的話，轉折點的個數總和會較大，若聲音是 voiced sound，轉折點的個數總和會較小。

關鍵字：清音與濁音、隨機性測試、音訊信號處理

Abstract

The purpose of this experiment is to analyze the degree of randomness of different segments in the audio signal, and determine which segments are random signals or not, and finally discuss the relationship between random signals and the unvoiced and voiced features in speech, and will also discuss the relationship between the length of the extracted segment and the judgment of random signals.

This experiment is based on the method on page 116 of the textbook. First, mark all the turning points in the audio signal, and then count the number of turning points in the segment by segment, and finally use the proportion of the number of turning points in the partial segment of the audio signal to the segment to determine whether the signal of this segment is a random signal.

The test signal used in this experiment is two recordings, the audio content is "safety" and "Wu Yi-feng", the recording tool is a mobile phone (OPPO R11s), and the audio sampling frequency is 44100 Hz.

After observing the results, it can be found that the characteristics are roughly consistent with the theory. If the sound is unvoiced sound and plosive sound, the total number of turning points will be larger. If the sound is voiced sound, the total number of turning points will be smaller.

Keyword：Unvoiced and voiced sound, Randomness test, Audio signal processing

目　次

一、 說 明

1. 在音訊信號中，語音主要可以分為清音(unvoiced sound)、濁音(voiced sound)和爆裂音(plosive sounds)。

   清音 ： 使空氣流通過聲道特定位置形成的狹窄開口時會產生，聲音的產生與聲道沒有關係，所以發出清音時喉嚨不會震動。

   濁音 ： 空氣流通過聲帶，使聲帶產生振盪時會產生，聲音的產生與聲帶有關係，所以發出濁音時喉嚨會震動。

   爆裂音 ： 聲道完全關閉，然後突然釋放積聚的壓力時會產生，因為太短暫了，所以很難標示爆裂音的特徵，此外，它們的特性也會受到進行中的音素所影響。(音素 ： 人類語言中能夠區別意義的最小聲音單位)。

2. 峰或谷由三個信號點所構成，若中心點比兩邊還高，稱為波峰；若中心點比兩邊還低，稱為波谷。

3. 轉折點(turning points)指的就是波峰和波谷。

4. 給定一段固定長度的信號樣本，假設為 N 個信號點。若信號中，轉折點的個數大於 $\frac{2}{3}(N-2)$，則該段信號可被標記為隨機樣本。

二、 程 式 碼

主要程式碼分為標記波峰波谷的函式計算長度固定的移動窗口中的波峰波谷個數與兩部分。

標記波峰波谷的函式有三個版本 :

< 標記波峰波谷的函式 (第一版) (會將平面的最後一個點視為波峰或波谷) >
輸入 : 聲音信號
輸出 : 與輸入信號等長，只有 0 和 1 的陣列，1 表示為波峰或波谷。

```
import numpy as np
# 標記波峰波谷
def mark_peak_ver1(sigs):
    # 計算斜率放入 diff，並將正斜率設為 1、負斜率設為 -1、斜率為零就是 0
    diff = np.diff(sigs)
    diff[diff > 0] = 1
    diff[diff < 0] = -1
    # 將 diff 前後補 0，再將後減前的結果放入 mark，此時 mark 中波峰位置的值會為 -2，波谷位置的值會為 2，平面的位置的值會為(-1,0,1)。
    # 平面還需要額外判斷，所以先只保留波峰波谷的值，將 mark 做 abs 再整除 2，此時波峰波谷位置的值會為 1，其他位置為 0。
#       mark = np.concatenate(([0], diff)) - np.concatenate((diff, [0]))
    mark = np.concatenate(([0], diff, [0]))
    mark = np.diff(mark)
    mark = abs(mark)
#       mark[mark < 2] = 0
    mark = mark // 2
    # 處理平面(斜率為 0)，若平面為波峰或波谷，則將平面的最後一點視為極值。
    # 重新計算斜率，並記錄平面前後的斜率，若平面前後斜率為一正一負，就把平面最後一個點位置的值改為 1。
    diff = np.diff(sigs)
    diff = np.concatenate(([0], diff))
    flag = False
    start_slope = 0
    end_slope = 0
    start_index = 0
    end_index = 0
    for i in range(len(diff)):
```

```
            if diff[i] == 0:
                if flag == False:
                    if diff[i - 1] > 0:
                        start_slope = 1
                    else:
                        start_slope = 0
                    start_index = i - 1
                    flag = True
            else:
                if flag == True:
                    if diff[i] > 0:
                        end_slope = 1
                    else:
                        end_slope = 0
                    end_index = i
                    if (start_slope == 1 and end_slope == 0) or (start_slope == 0
and end_slope == 1):
#                           mark[start_index : end_index] = 1
                        mark[i - 1] = 1
                    flag = False
        # 信號的開始跟結束的點不考慮，直接設為 0
        mark[0] = 0
        mark[-1] = 0
        return mark
```

< 標記波峰波谷的函式 (第二版) (不將平面視為波峰或波谷) >
輸入 : 聲音信號
輸出 : 與輸入信號等長，只有 0 和 1 的陣列，1 表示為波峰或波谷。

```
import numpy as np
# 標記波峰波谷
def mark_peak_ver2(sigs):
    import scipy
    # local_max 為局部波峰的 index 陣列，local_min 為局部波谷的 index
陣列
    local_max = scipy.signal.argrelextrema(sigs, np.greater)[0]
    local_min = scipy.signal.argrelextrema(sigs, np.less)[0]
    # 將波峰波谷的 index 陣列合併後排序，放入 mark
    mark = np.concatenate((local_max, local_min))
```

```
    mark = np.sort(mark)
    # 宣告一個長度跟輸入信號相等的零陣列，並將波峰與波谷的地方改為1
    temp = np.zeros(len(sigs))
    temp[mark] = 1
    return temp
```

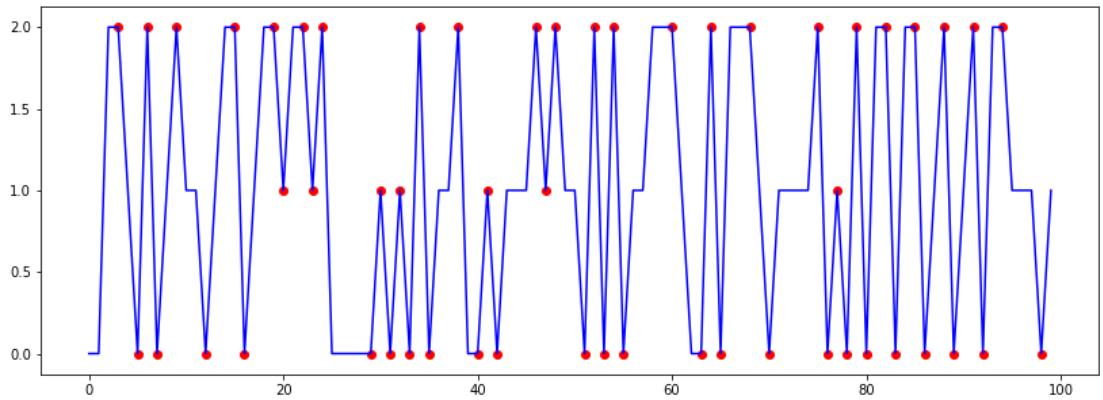
< 標記波峰波谷的函式 (第三版) (會將平面的每個點視為波峰或波谷) >
輸入 : 聲音信號
輸出 : 與輸入信號等長，只有 0 和 1 的陣列，1 表示為波峰或波谷。

```
import numpy as np
# 標記波峰波谷
def mark_peak_ver3(sigs):
    # 計算斜率放入 diff，並將正斜率設為 1、負斜率設為 -1、斜率為零就是 0
    diff = np.diff(sigs)
    diff[diff > 0] = 1
    diff[diff < 0] = -1
    # 將 diff 前後補 0，再將後減前的結果放入 mark，此時 mark 中波峰位置的值會為 -2，波谷位置的值會為 2，平面的位置的值會為(-1,0,1)。
    # 平面還需要額外判斷，所以先只保留波峰波谷的值，將 mark 做 abs 再整除 2，此時波峰波谷位置的值會為 1，其他位置為 0。
#       mark = np.concatenate(([0], diff)) - np.concatenate((diff, [0]))
    mark = np.concatenate(([0], diff, [0]))
    mark = np.diff(mark)
    mark = abs(mark)
#       mark[mark < 2] = 0
    mark = mark // 2
    # 處理平面(斜率為 0)，若平面為波峰或波谷，則將平面的每個點視為極值。
    # 重新計算斜率，並記錄平面前後的斜率，若平面前後斜率為一正一負，就把平面的每個點位置的值改為 1。
    diff = np.diff(sigs)
    diff = np.concatenate(([0], diff))
    flag = False
    start_slope = 0
    end_slope = 0
    start_index = 0
```

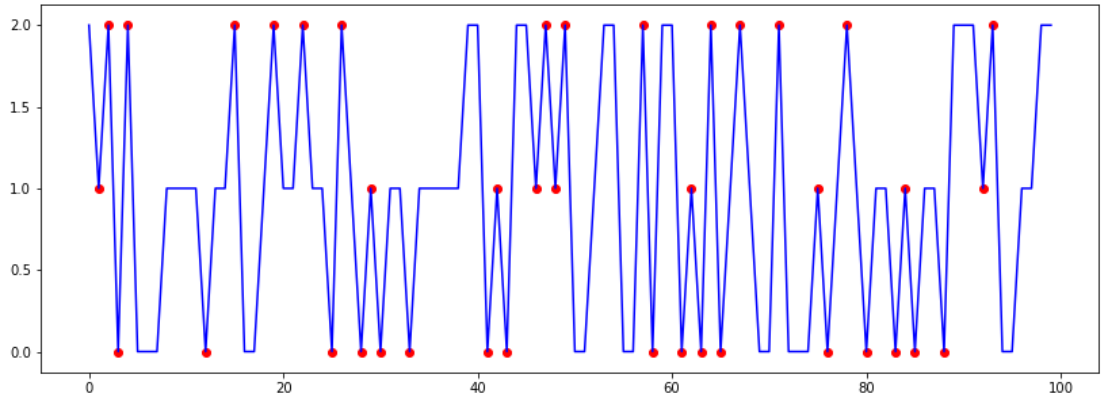
```
        end_index = 0
        for i in range(len(diff)):
            if diff[i] == 0:
                if flag == False:
                    if diff[i - 1] > 0:
                        start_slope = 1
                    else:
                        start_slope = 0
                    start_index = i - 1
                    flag = True
            else:
                if flag == True:
                    if diff[i] > 0:
                        end_slope = 1
                    else:
                        end_slope = 0
                    end_index = i

                    if (start_slope == 1 and end_slope == 0) or (start_slope == 0
and end_slope == 1):
                        mark[start_index : end_index] = 1
                    flag = False
        # 信號的開始跟結束的點不考慮，直接設為 0
        mark[0] = 0
        mark[-1] = 0
        return mark
```

波峰波谷標記測試結果：

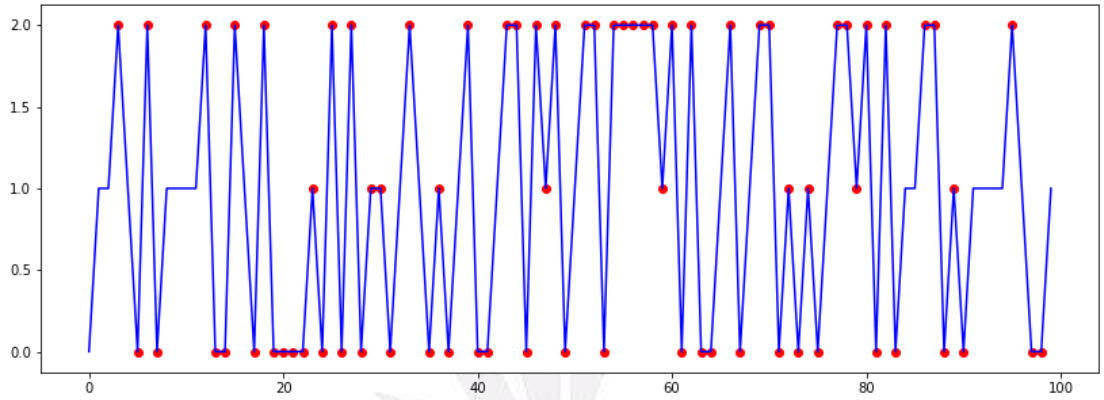
第一版 ：



第二版 ：



第三版 ：



計算長度固定的移動窗口中的波峰波谷個數有兩個版本 ：

< 計算長度固定的移動窗口中的波峰波谷個數(第一版) > 輸入 ：{ peak： 信號標記是否為波峰波谷的陣列 win_sec： 窗口的時間長度 sample_rate： 信號的採樣頻率 } 輸出 ： 信號中，往前一定時間內的波峰波谷個數和。

import numpy as np
# 計算一定信號長度內的頂點個數 	
def count_peak_ver1(peak, win_sec, sample_rate):
# 計算窗口的長度 	
win_len = int(win_sec * sample_rate)
# 將 peak 作累積和 	
# 將累積和從 win_len 到結束的值，減掉前 win_len 點的值 	
peak_sum = peak.cumsum()
peak_sum[win_len:] = peak_sum[win_len:] - peak_sum[:-win_len]
return peak_sum

```
< 計算長度固定的移動窗口中的波峰波谷個數( 第二版 ) >
輸入 :{ peak : 信號標記是否為波峰波谷的陣列
        win_sec : 窗口的時間長度
        sample_rate : 信號的採樣頻率 }
輸出 : 信號中，往前一定時間內的波峰波谷個數和。
```

```
import numpy as np
# 計算一定信號長度內的頂點個數
def count_peak_ver2(peak, win_sec, sample_rate):
    # 計算窗口的長度
    win_len = int(win_sec * sample_rate)
    # 計算不同時間時窗口內的波峰波谷總數。
    # 因為前幾秒的長度還不夠，所以值都為前 win_sec 秒的總和，超過 win_sec 秒後窗口才會開始移動。
    peak_sum = []
    for i in range(len(peak)):
        if i < win_len:
            peak_sum.append(sum(peak[:win_len]))
        else:
            peak_sum.append(sum(peak[i - win_len : i]))
    return np.array(peak_sum)
```

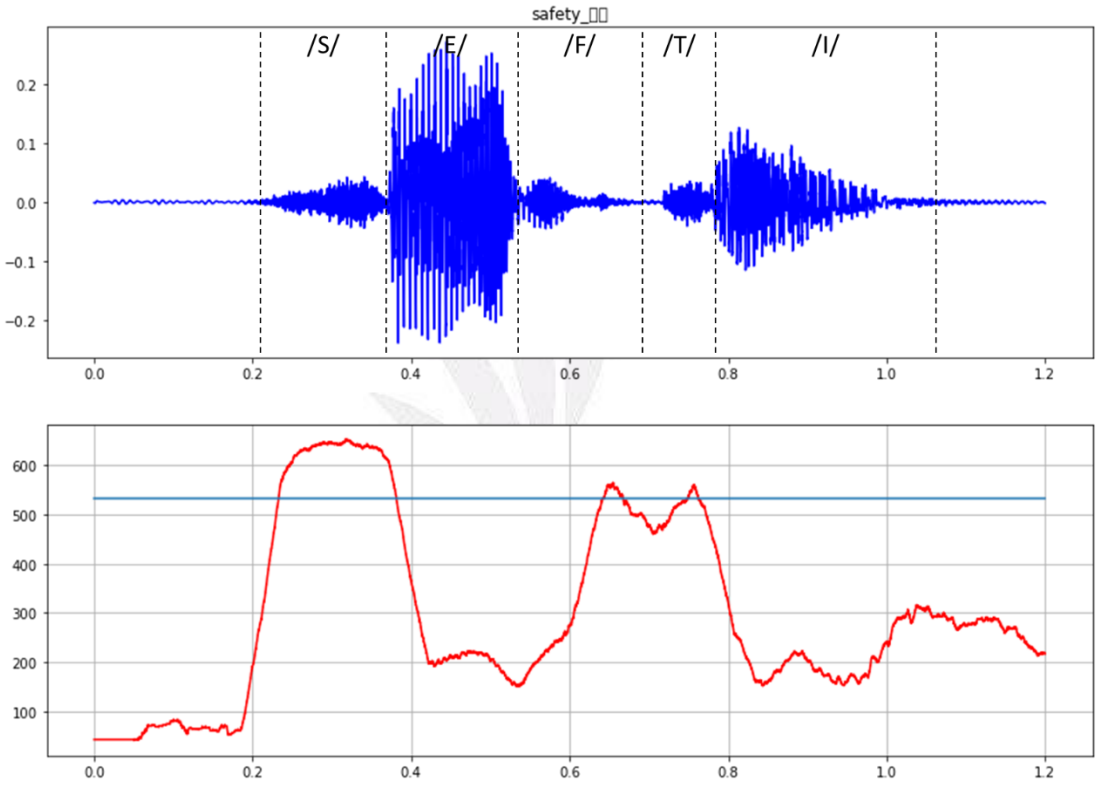
三、 實 驗 結 果

■ 使用參數 ：

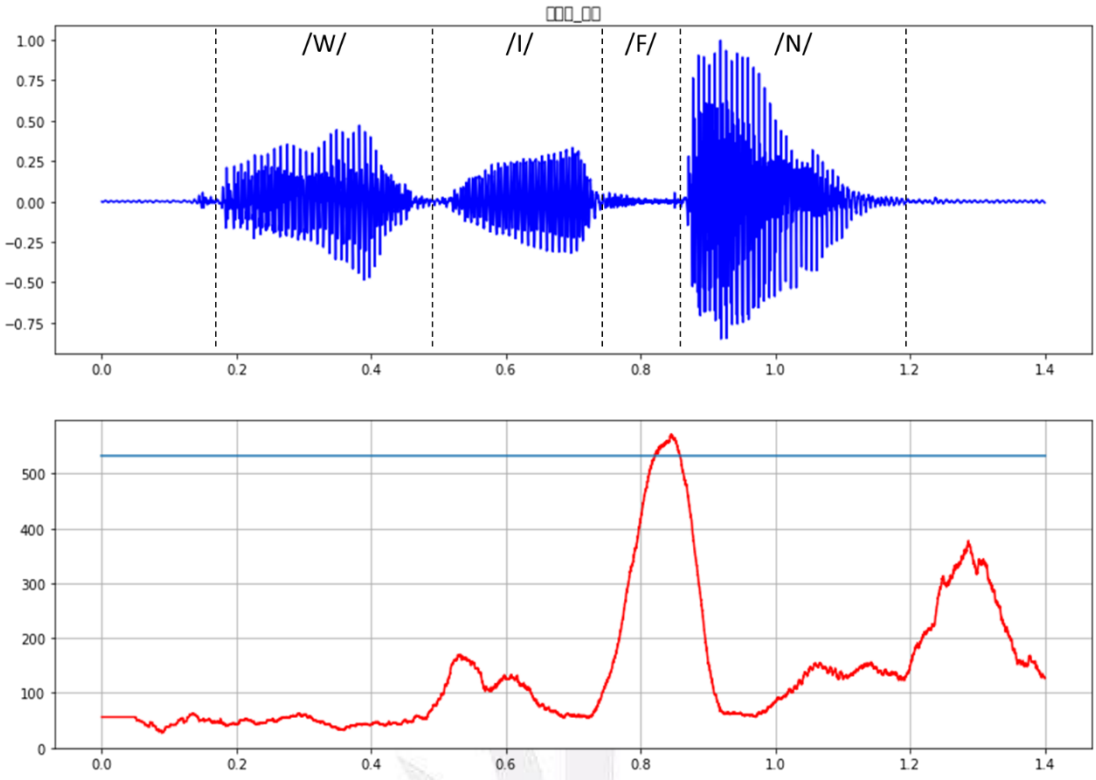
因為用 44100 Hz 與 8000 Hz 的採樣頻率將音檔讀出後，峰值總和的特徵與課本的不太一樣，而 16000 Hz 的結果比較接近理論結果，所以這裡就使用 16000 Hz 把音檔讀出來。

採用第一版的標記波峰波谷方式，再依照講義 117 頁的參數，把 window 的時間設為 0.05 秒，判斷是否為隨機信號的閾值為 $\frac{2}{3}(0.05 \cdot 16000 - 2) = 532$。

音檔內容 ：Safety



音檔內容 ： 吳宜峰



■ 結論 ：

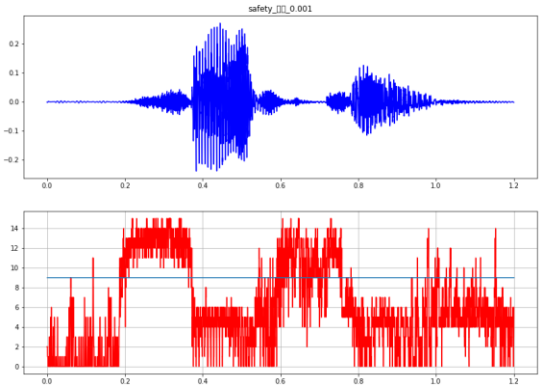
上圖為兩個音檔的分析結果，圖分成兩個部分，上半部是音訊信號，並標示當時是什麼音，下半部為閾值(藍線)與轉折點總和結果(紅線)，觀察後可以發現特徵跟理論大致有符合。聲音如果是 unvoiced sound 與 plosive sound 的話，轉折點的個數總和會較大，若聲音是 voiced sound，轉折點的個數總和會較小。

但是我的音檔在沒有聲音的時候會有雜音，轉折點的個數總和會忽大忽小

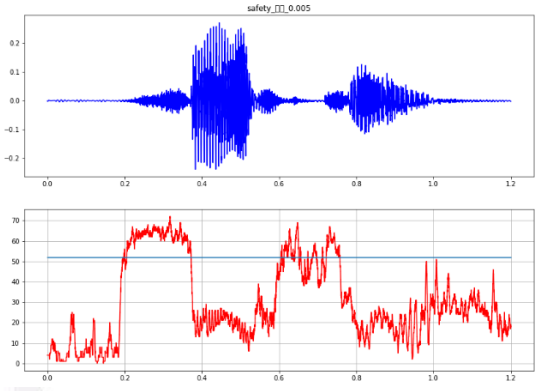
四、 額 外 問 題

最後要額外探討改變 Window 長度會產生什麼樣的影響？如果希望這種語音測試的演算法可以自動化進行，你(妳)覺得有什麼可行的方法？

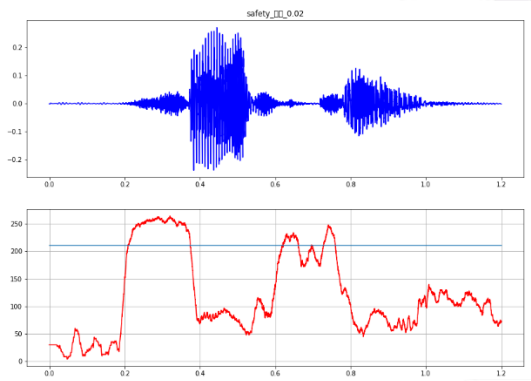
■ **實驗結果 ：**



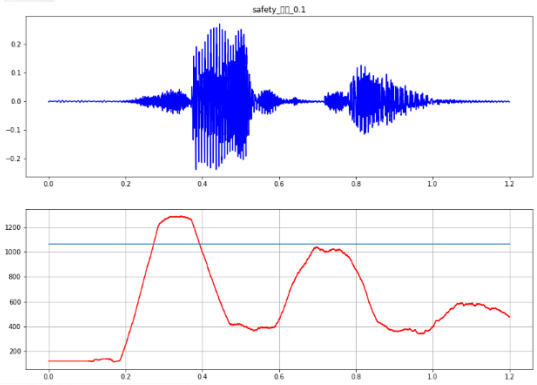
⇧ Window_len = 0.001 sec



⇧ Window_len = 0.005 sec



⇧ Window_len = 0.02 sec



⇧ Window_len = 0.1 sec

■ 結論 ：

Window 長度越短，總和的結果反應會越快速、越貼近聲音改變的時間點，但是也會越不平滑，所以可能會很容易不小心就超過閾值。若 window 長度越長則相反。

若 window 長度大於不同聲音的最小持續時間，則可能會造成 unvoiced sound 與 plosive sound 發生時，轉折點的個數總和被左右平均掉，導致結果沒辦法超過閾值。

我認為 window 長度可以從很短開始測試，再讓程式自動去檢查，若高於閾值的部分有很多持續時間都很短，就再延長 window 的長度，直到會超過閾值的部分的持續時間都不會太短就可以固定 window 長度。

五、 參 考 文 獻

[1] R. M. Rangayyan, Biomedical Signal Analysis, 2nd Ed., Wiley, 2015.