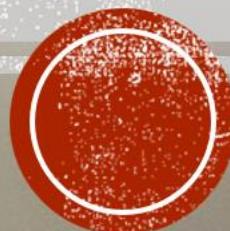


MUSIC TRANSCRIPTION BY PYTHON

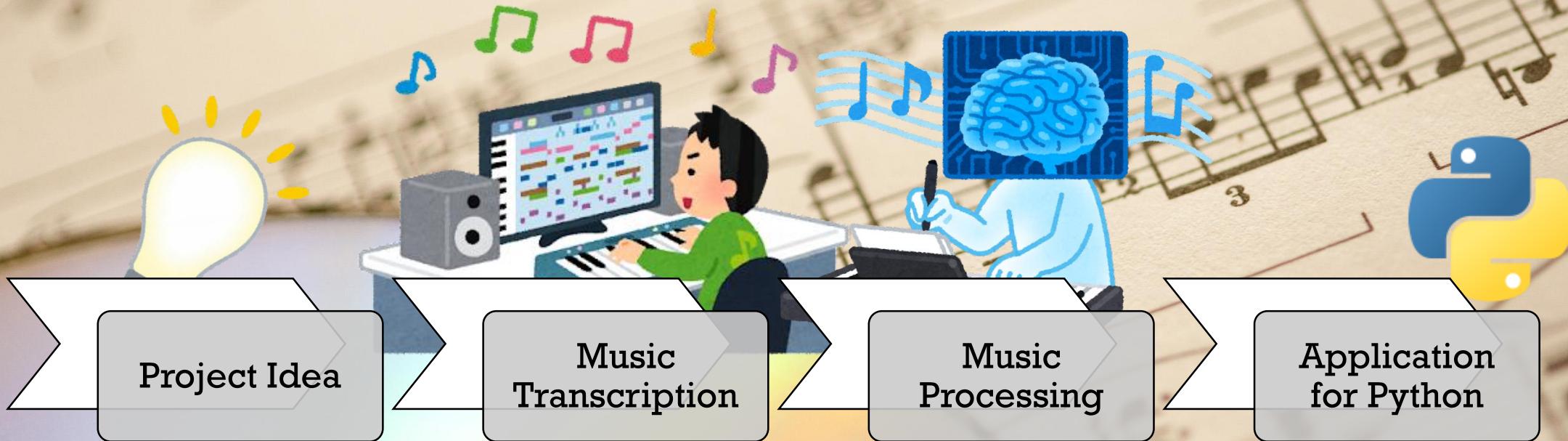
期末專案 12

小組成員：

張程翔 汪晁安 耿睿棋



OUTLINE



PROJECT IDEA



- Listen to music

- Wonder how to play it

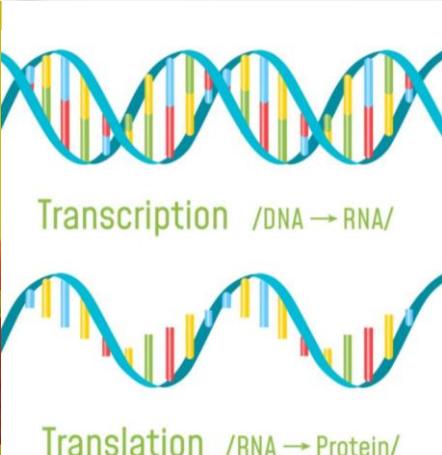
- Using Python as a tool





MUSIC TRANSCRIPTION

What is transcription?



採譜
(transcription)
亦稱轉譜，指的是
透過人耳對一首樂
曲的反覆試聽，從
而將其總譜寫下來
的過程。

How to transcript music?



MUSIC PROCESSING

A frontiers for us to explore

High-level mathematics applied

Music theory? What's that?



HIGH-LEVEL MATHEMATICS?

by the sampling period T . As a result, one obtains the following approximation:

Don't forget some matrices!

$$\text{DFT}_N \cdot \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{pmatrix} = \begin{pmatrix} \text{id}_M & \Delta_M \\ \text{id}_M & -\Delta_M \end{pmatrix} \left(\frac{\text{DF}}{C} \right)$$

2.1.1

Fourier

Based on the similarity measure (2.3), we compare the signals $g = \cos_{\omega, \varphi}$ as defined in (2.2). For a fixed frequency

$$d_\omega := \max_{\varphi \in [0,1]} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega, \varphi}(t) dt \right)$$

$$\varphi_\omega := \operatorname{argmax}_{\varphi \in [0,1]} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega, \varphi}(t) dt \right)$$

As previously discussed, the magnitude coefficient

frequency ω within the signal f . Additionally, the phase coefficient $\varphi_\omega \in [0, 1]$



$$x p(-2\pi i \omega n T) \approx \hat{f}(\omega). \quad (2.21)$$

Riemann sum

be interpreted as the overall area of rectangular corresponding to the integral (see Figure 2.7d). ωn as a **Riemann sum**. As we will show in roximation is good for “well-behaved” signals ω .

If the system is in state α_i at time t , and it jumps into the state α_{i_1} (with probability a_{i_1}), then it moves on to state α_{i_2} (with probability $a_{i_1 i_2}$), and

'bout some probabilities?

$$a_{i_1} \cdot a_{i_1 i_2} \cdot b_{i_2 k_2} \cdot \dots \cdot a_{i_{N-1} i_N} \cdot b_{i_N k_N}. \quad (5.32)$$

ility $P[O | \Theta]$, one needs to consider all possible

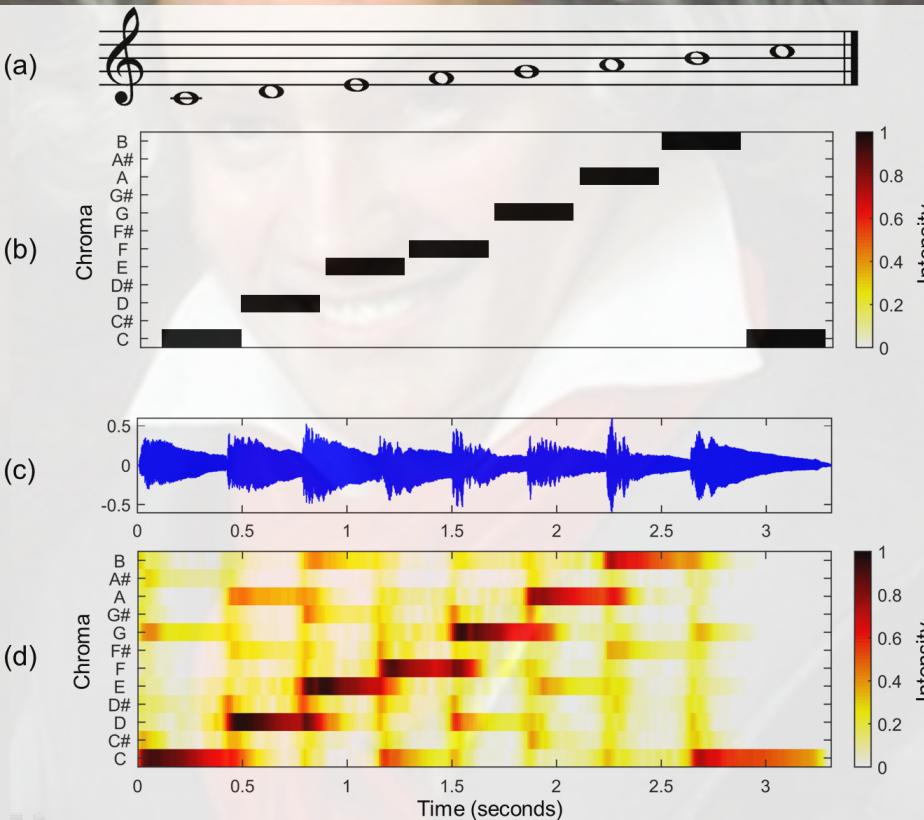
$$[O, S | \Theta] \quad (5.33)$$

$$= \sum_{i_1=1} \sum_{i_2=1} \dots \sum_{i_N=1} c_{i_1} \cdot b_{i_1 k_1} \cdot a_{i_1 i_2} \cdot b_{i_2 k_2} \cdot \dots \cdot a_{i_{N-1} i_N} \cdot b_{i_N k_N}. \quad (5.34)$$





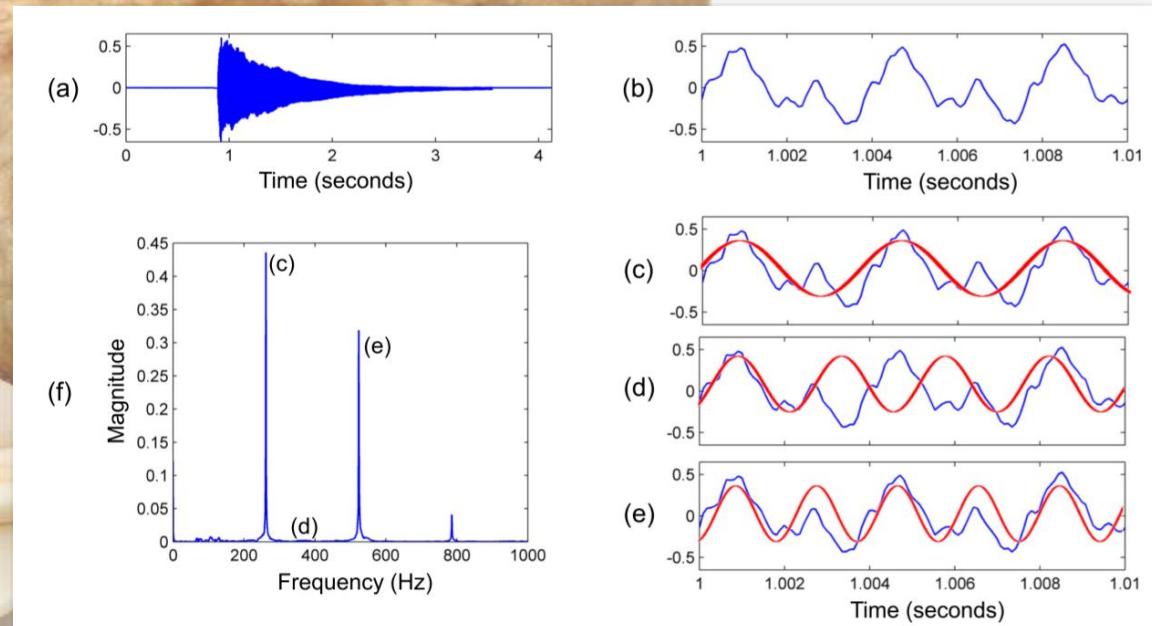
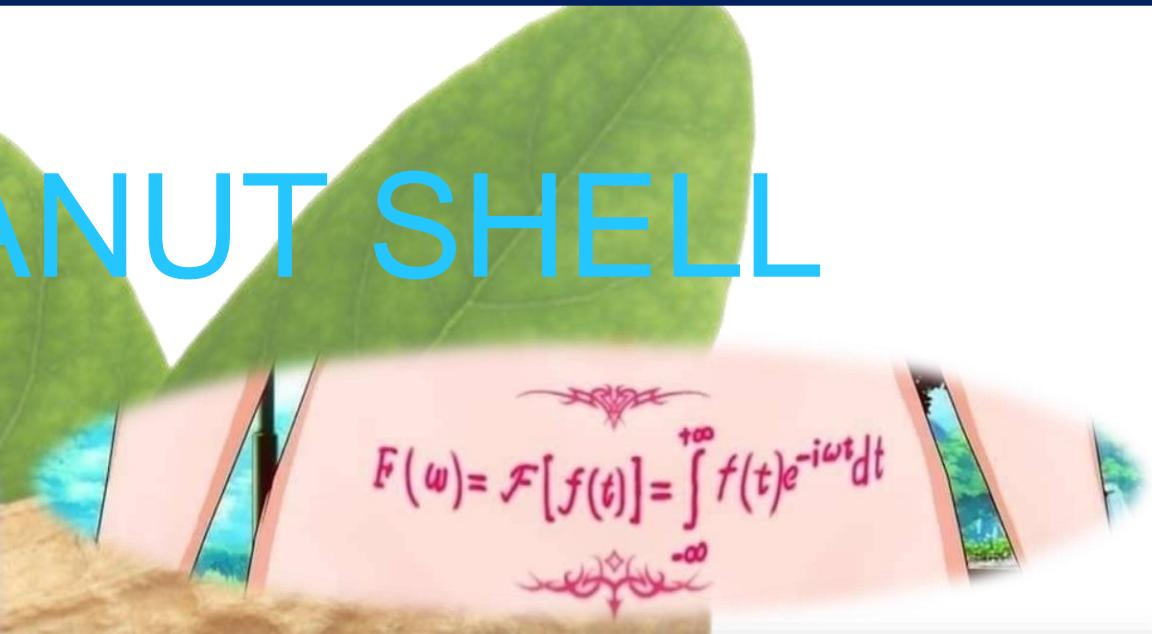
MUSIC THEORY



HOW DO WE OVERCOME IT?



FT IN A PEANUT SHELL



PRACTICAL APPLICATIONS





SAMPLE A



卿雲歌

Andante maestoso

卿
卿雲爛兮亂漫漫



SAMPLE B



Dance Of the Sugar Plum Fairy

Transcribed by Nohpets
Tchaikovsky

$\text{J} = 70$

The sheet music consists of three staves of musical notation. The top staff is for the treble clef (G-clef) voice, and the bottom staff is for the bass clef (F-clef) voice. The key signature is one sharp (F#). The tempo is marked as $\text{J} = 70$. The first measure shows a dynamic of pp (pianissimo). The music features a repetitive eighth-note pattern with various dynamics including mf (mezzo-forte), f (forte), and p (piano). The notation includes several grace notes and sixteenth-note patterns.



ADDITIONAL-SAMPLE G



Handwritten musical score for 'Giant Steps' by John Coltrane. The score is written on four staves of five-line music staff paper. The tempo is marked as 'FAST' and '170.'. The title 'GIANT STEPS' is written above the first staff. The score includes various chords and notes, with some markings like 'B' and 'F#'. The right side of the page shows a portion of a vinyl record.



ADDITIONAL-SAMPLE G'



Track Name Type On Bus Out Device Channel Instrument V
✓ Track 1 Instrument on General MIDI - Microso 1 Acoustic Grand

Style Treble + Bass Staff Lyrics line 1
Key C Major - no grid --
Time 4/4 Zoom 100% Lyrics Font...
Show 16th notes Show Tracks... Harmonize...
 Lock Score Insert Mode 50

Dotted
Duplet
Triplet
Quintuplet
Septuplet
Staccato

Insert Rest ...



PROBLEMS WHEN ANALYZING

Too much information...

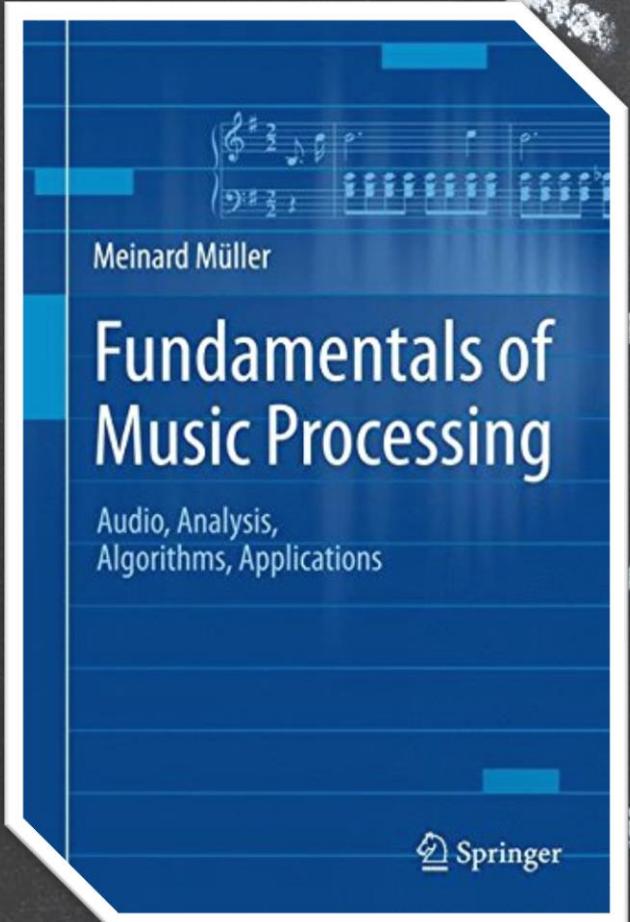
How to filter noise figure?

How to present?





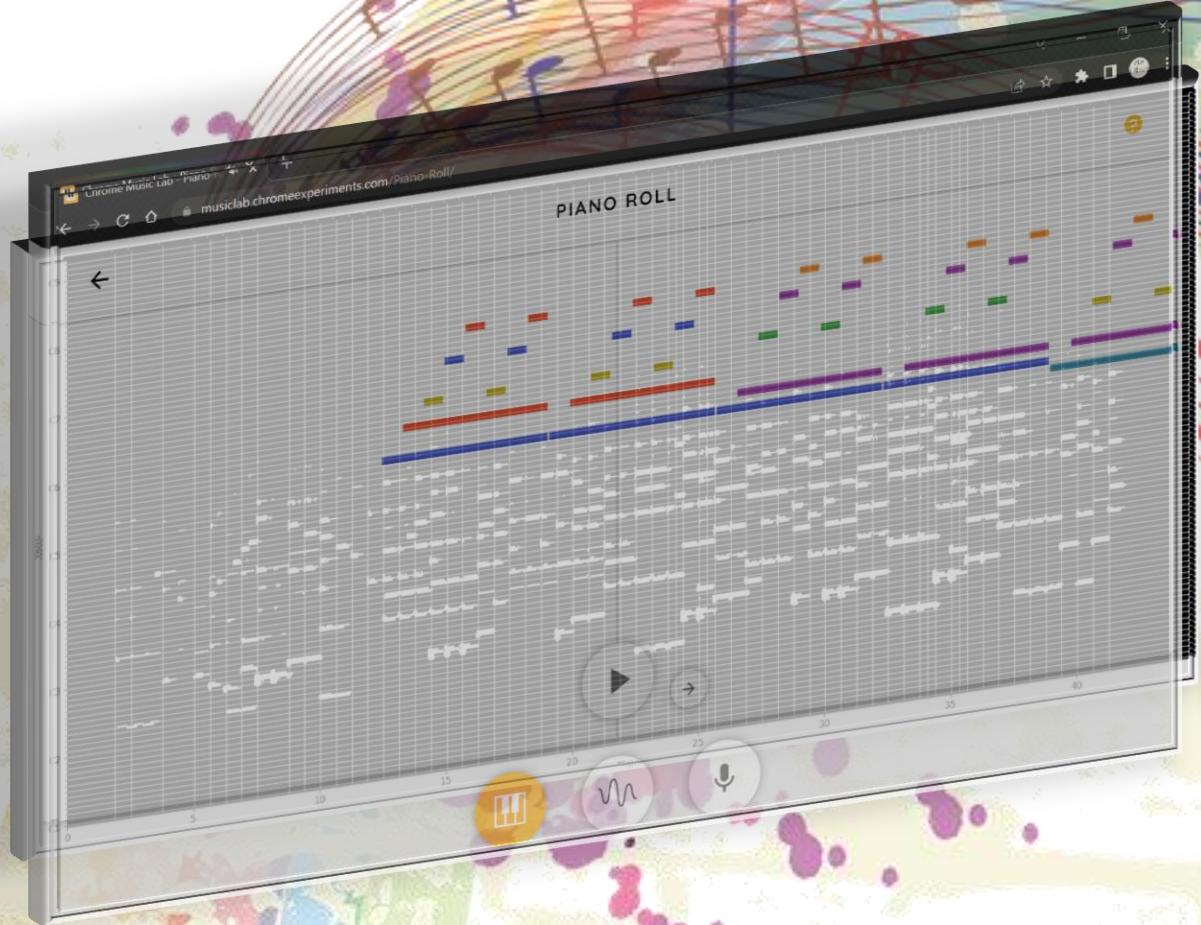
TOO MUCH INFORMATION



What does this project need?
Theorem? Math? HMM? MIDI?



HOW TO PRESENT?



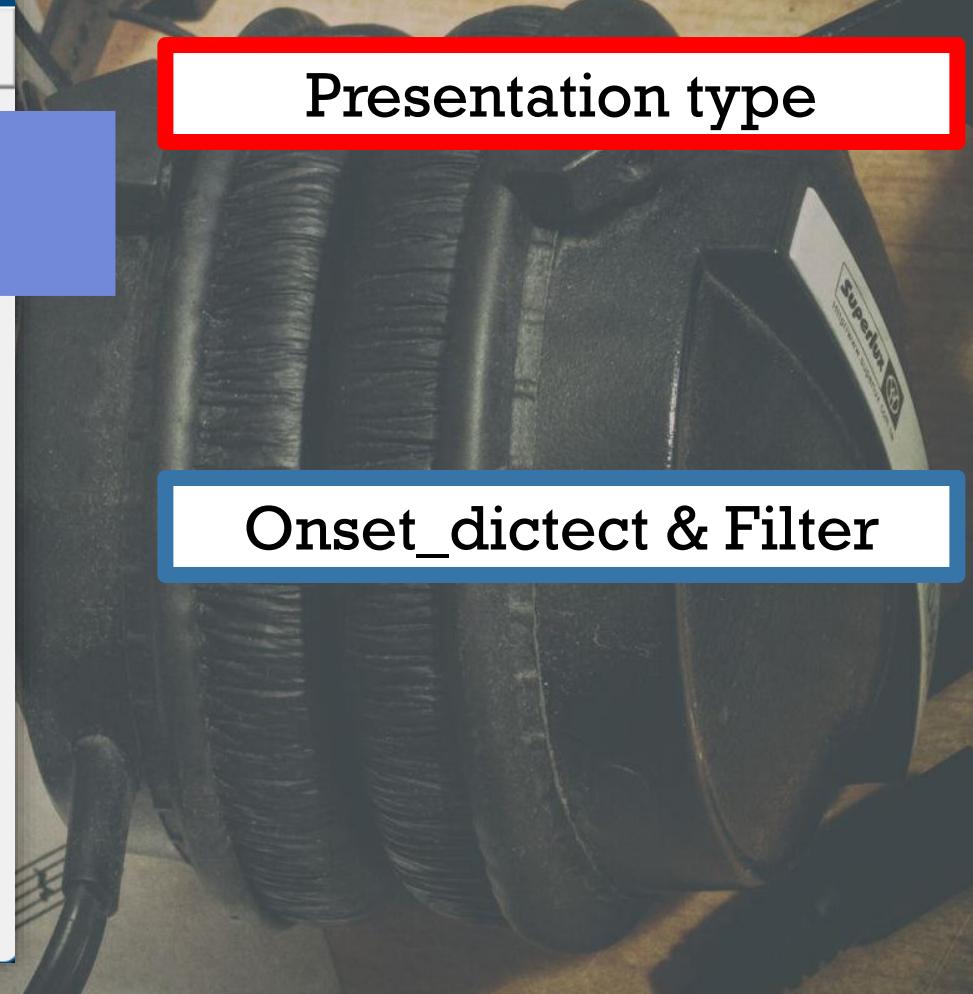
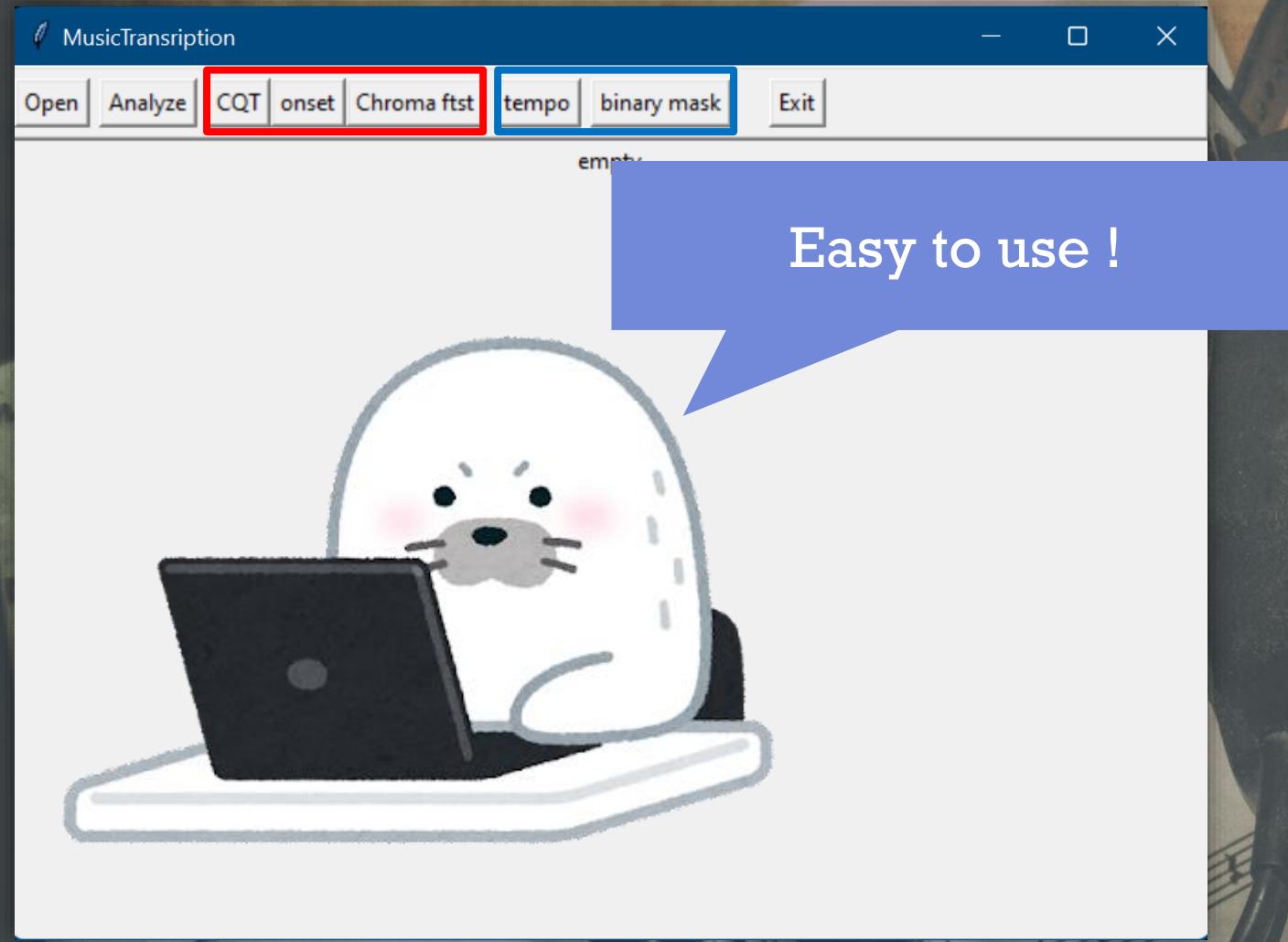


.MP3 VS .WAV

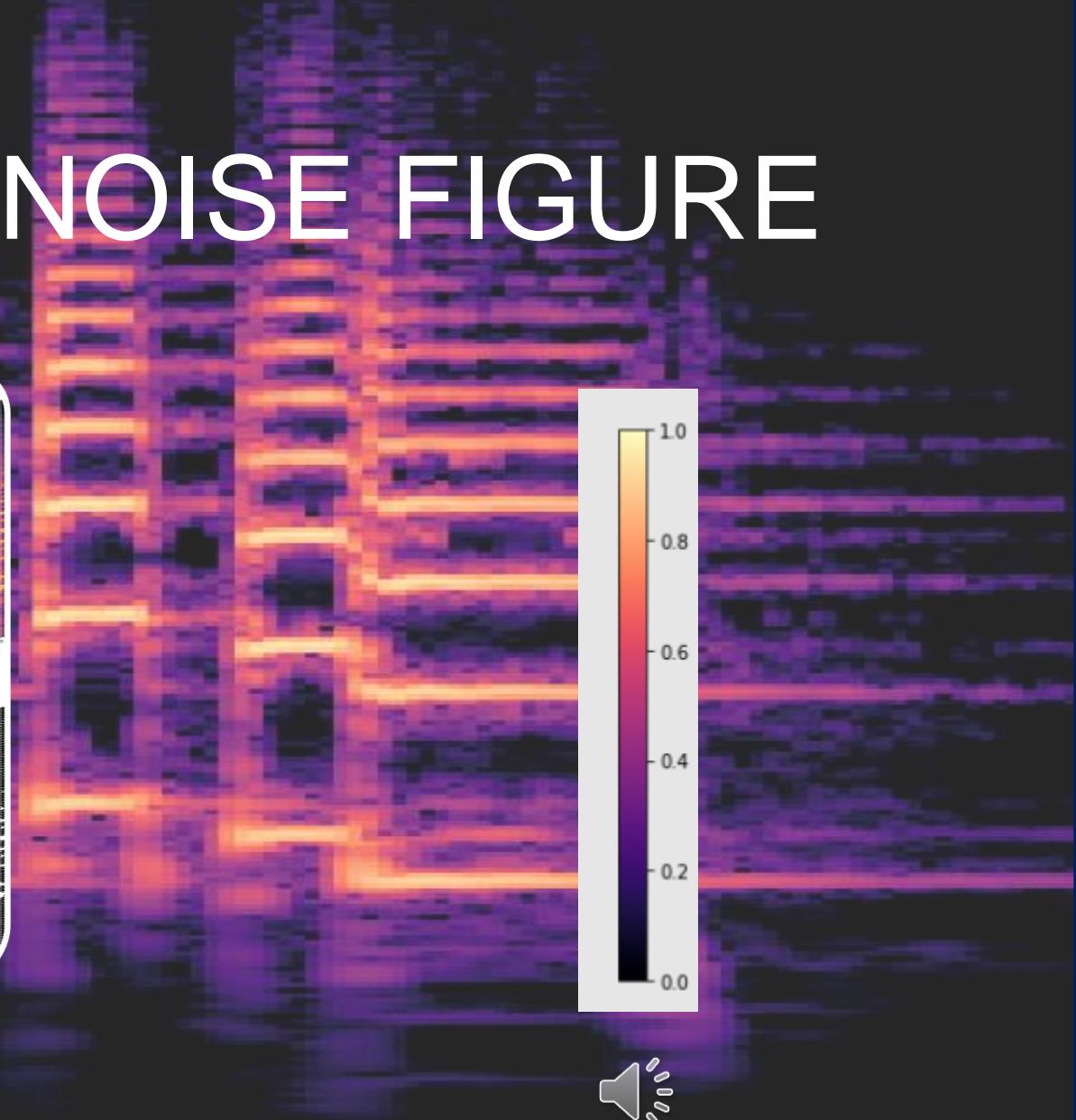
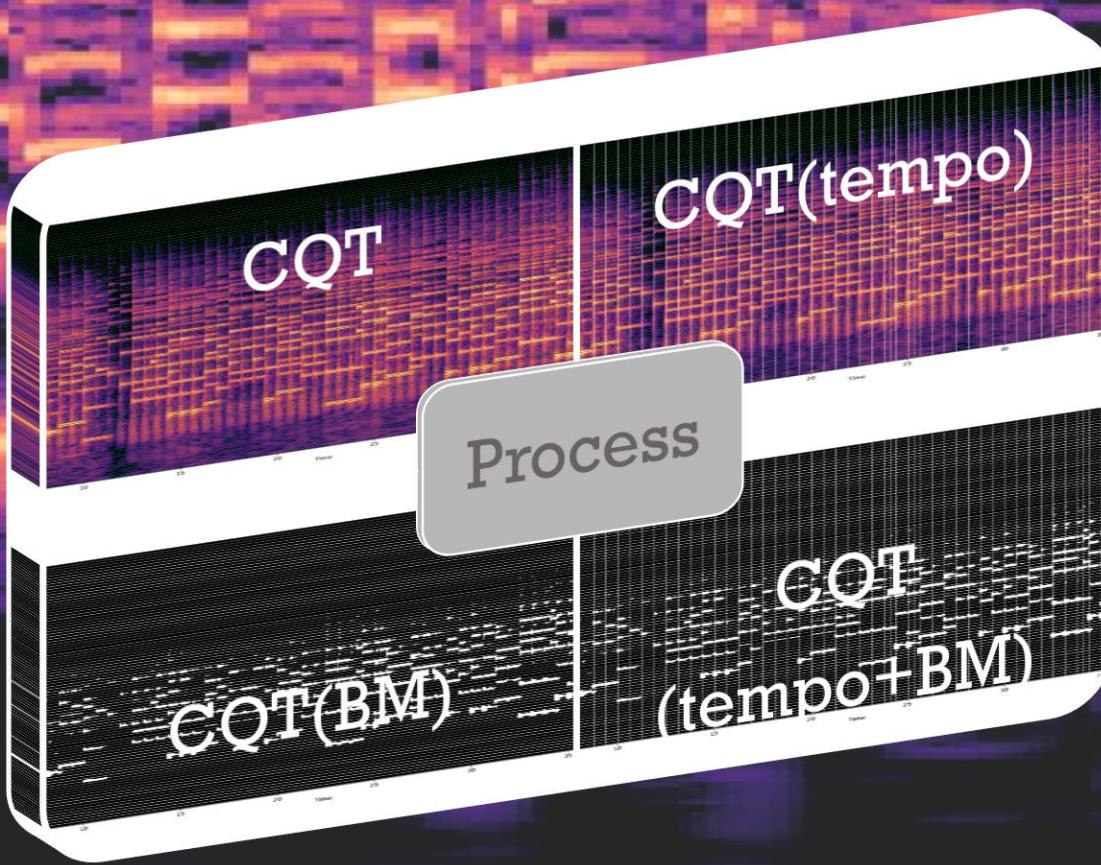




GUI



HOW TO FILTER NOISE FIGURE



PERFORMANCE SAMPLE A

卿雲歌

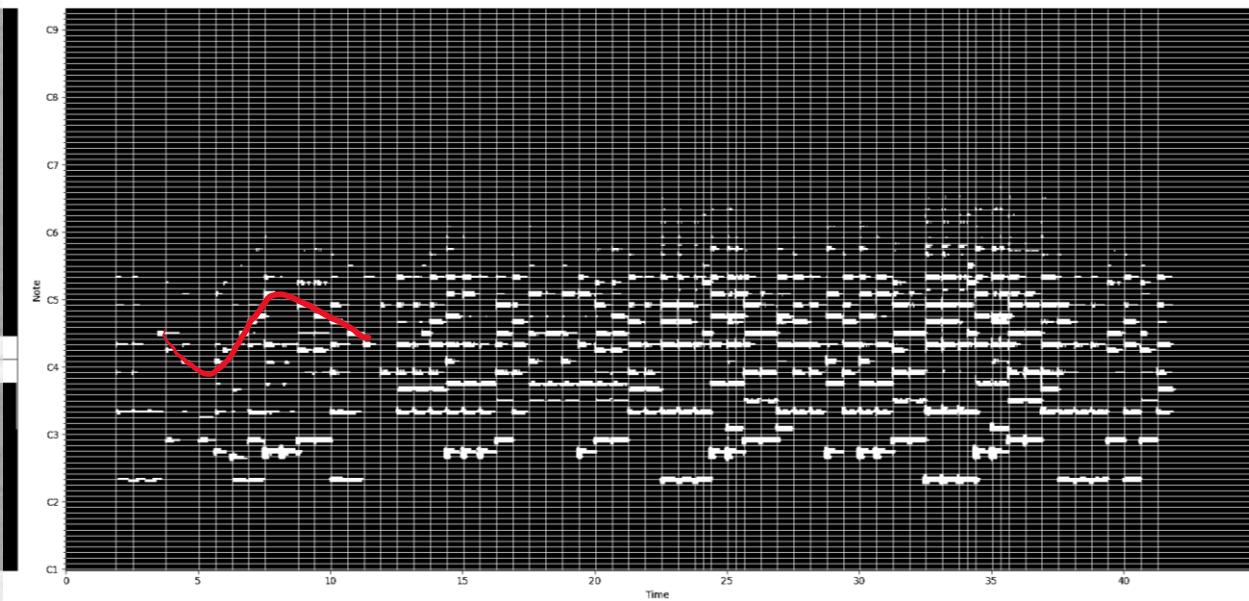
Andante maestoso

Pitch class
B
A
G
F
E
D
C
0

Time
40 30 20 10 0

雲 煙 此 境 境 分

mf



PERFORMANCE SAMPLE B

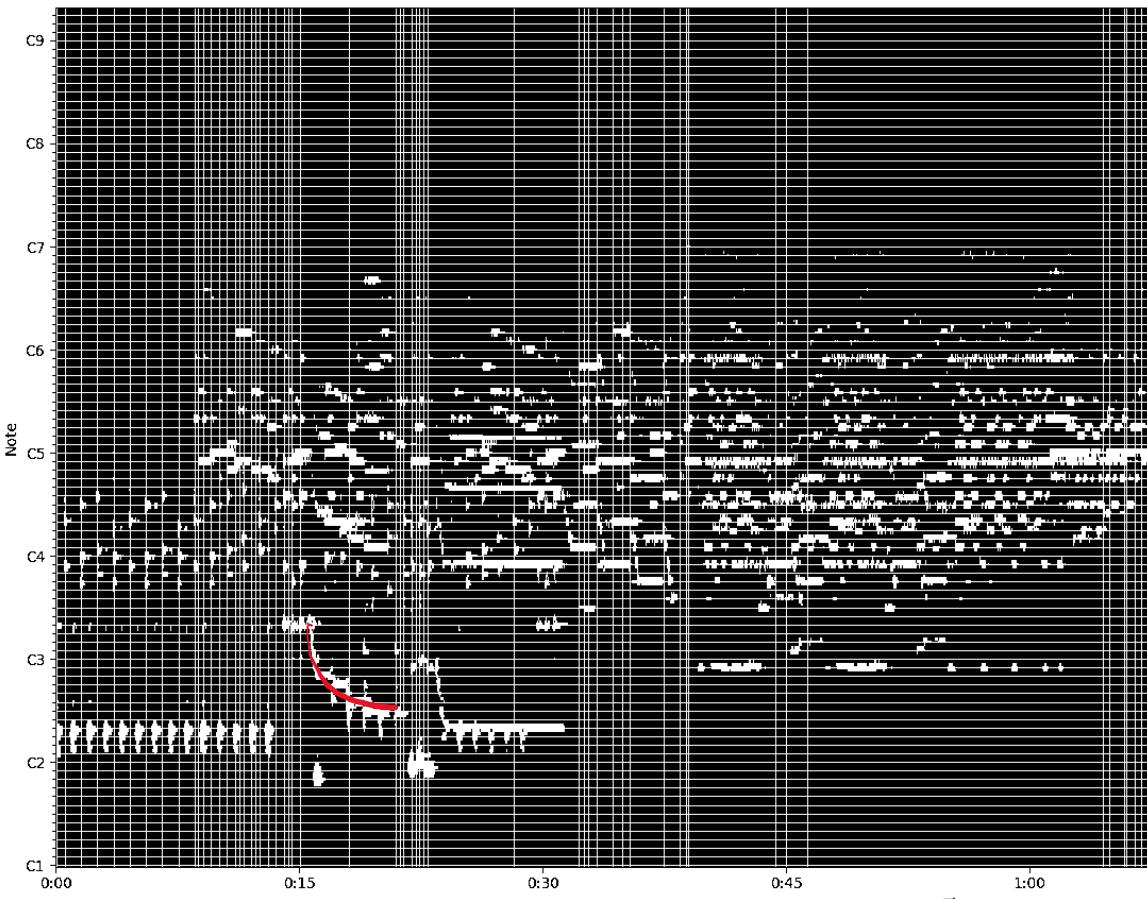
Dance Of the Sugar Plum Fairy

Transcribed by Nohpets
Tchaikovsky

$\text{J} = 70$

6

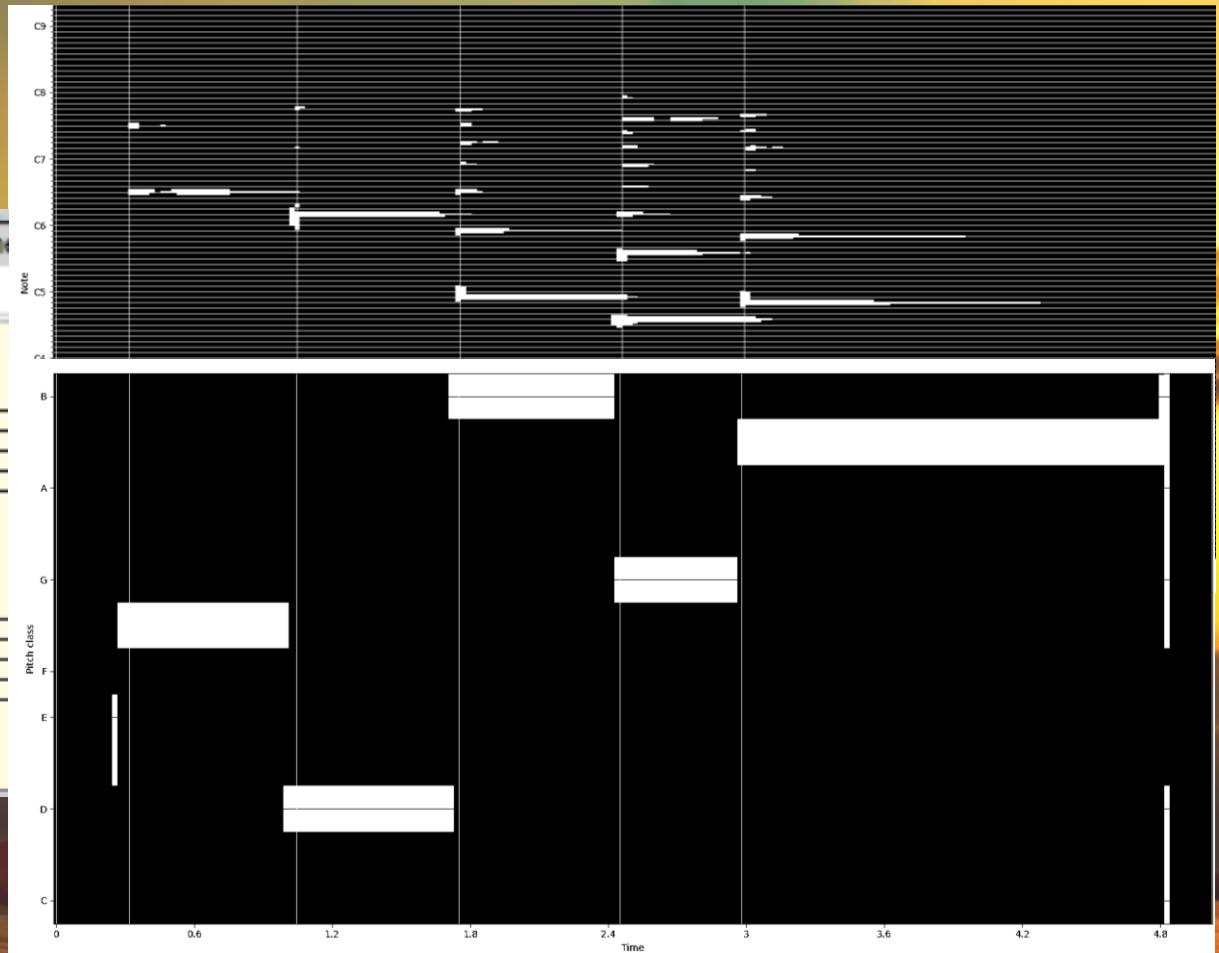
10



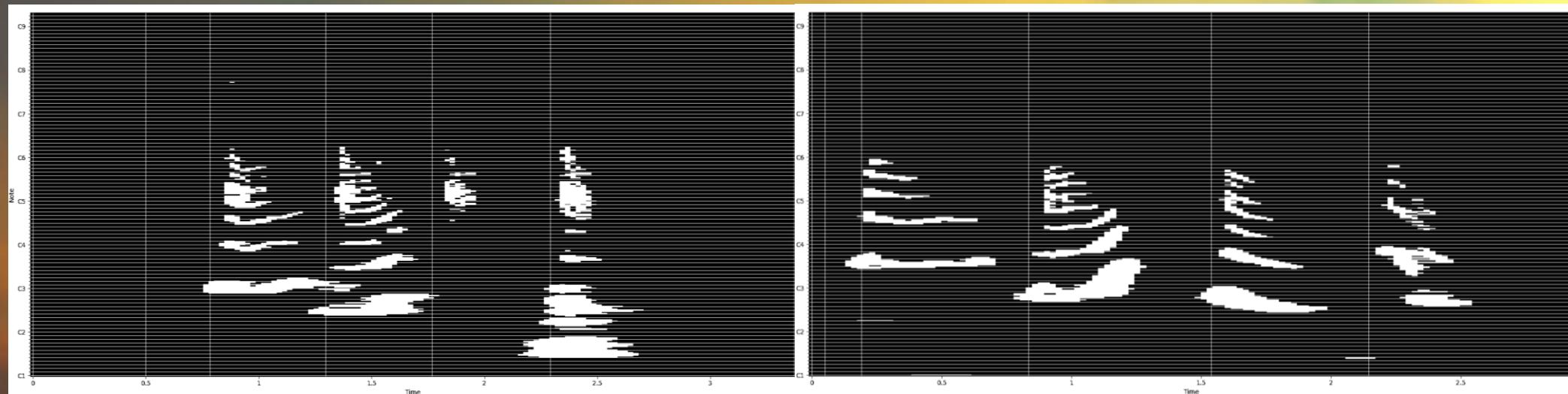
PERFORMANCE SAMPLE G'

Track Name	Type	On	Bus Out	Device	Channel
Track 1	Instrument	on		General MIDI - Microso	1

MIDI interface showing Track 1 settings and two musical staves. The top staff is in treble clef, 4/4 time, with notes E, B, G, D, A, E. The bottom staff is in bass clef, 4/4 time, with notes E, B, G, D.

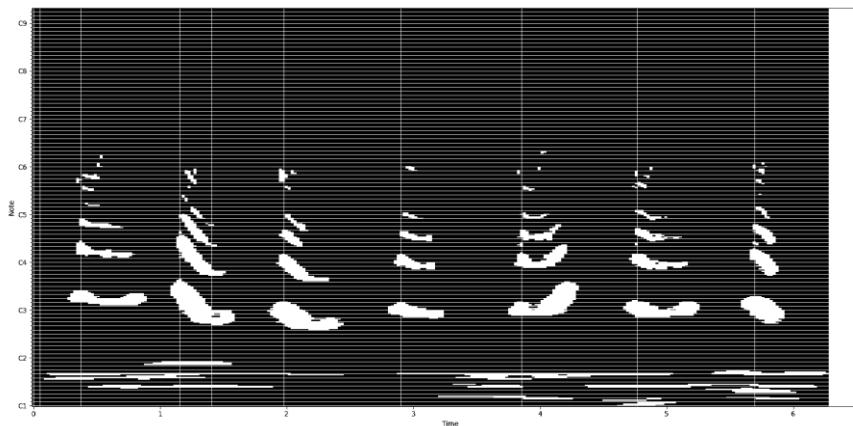


APPLICATION FOR TONES ANALYZING

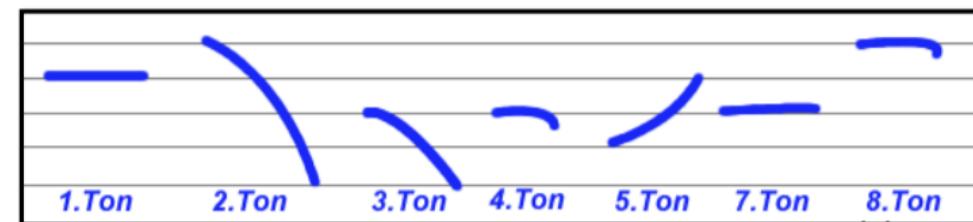


Different people say: “通同統痛”

APPLICATION FOR TONES ANALYZING



Tone Types	君 [kun˥]	滾 [kun˥˥]	棍 [kun˧˥]	骨 [kut˥˧]	裙 [kun˩]	-	近 [kun˧˨]	滑 [kut˥˧]
Tone Number	44	53	21	3	12		22	5
IPA	˥	˥˥	˧˥	·	˩		˧˨	˥˧



(右圖來自江文瑜教授對臺語音調所提供的參考資料)



[Notification]
You got some codes!



Implementation

```
#載入套件
print("正在載入套件。")
import os
try:
    import ffmpeg
except:
    print("ffmpeg未安裝，正在下載")
    os.system('pip install ffmpeg')
    import ffmpeg
filename = "123.mp4"
if (filename + "\\bin") not in os.environ["path"]:
    os.environ["path"]+=(("\\bin"+filename+"\\bin"))
print("path added")#處理path
try:
    import tkinter
except:
    print("tkinter未安裝，正在下載")
    os.system('pip install tkinter')
    import tkinter
try:
    import numpy
except:
    print("numpy未安裝，正在下載")
    os.system('pip install numpy')
    import numpy
try:
    import matplotlib.pyplot as plt
except:
    print("matplotlib未安裝，正在下載")
    os.system('pip install matplotlib')
```

#IMPORT MODULES

```
import librosa, librosa.display
except:
    print("librosa未安裝，正在下載")
    os.system('pip install librosa')
    import librosa, librosa.display
try:
    import tkinter as tk
    from tkinter.filedialog import askopenfilename
except:
    print("tkinter未安裝，正在下載")
    os.system('pip install tkinter')
    import tkinter as tk
    from tkinter.filedialog import askopenfilename
try:
```

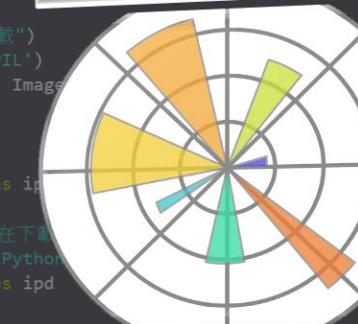
Numpy



NumPy

```
import ImageTk, Image
except:
    print("PIL未安裝，正在下載")
    os.system('pip install PIL')
    from PIL import ImageTk, Image
try:
    import IPython
except:
    print("IPython未安裝，正在下載")
    os.system('pip install IPython')
    import IPython.display as ipd
try:
    import scipy
except:
    print("scipy未安裝，正在下載")
    os.system('pip install scipy')
    import scipy
```

pyplot



GG

```
import IPython
except:
    print("IPython未安裝，正在下載")
    os.system('pip install IPython')
    import IPython
```

IP[y]

GG



GG



```
#定義按鈕指令
def open_file():
    global x; global sr
    filename = askopenfilename(
        filetypes = [ ("OGG Files", "*.ogg"), ("MP3 Files", "*.mp3"), ("WAV Files", "*.wav") ]
    )
    if not filename:
        return
    if filename[-4:]=="mp3":
        audSeg = pydub.AudioSegment.from_mp3(filename)
        audSeg.export(__file__[:-22] + filename.split("/")[-1][-4:] + ".wav", format="wav")
        filename = __file__[:-22] + filename.split("/")[-1][-4:] + ".wav"
        x, sr = librosa.load(filename[-4:] + ".wav")
        reply["text"] = "mp3格式不支援，將其轉換成wav。\\n" + filename
    else:
        x, sr = librosa.load(filename)
        reply["text"] = filename
def analyze():
    global x; global sr; global spec_image; global cur_img
    cur_img="spec_chromaSTFT.png"
    if sr == 0:
        return
    plt.rcParams['figure.figsize'] = (22, 10.5)
    bins_per_octave = 36
    #something here
```

#DEF ANALYZE

Reads file, and convert it to .wav

Analyze the audio file and produce the STFT graph, CQT graph, and onset graph with filters





#DEF ANALYZE

```
#定義按鈕指令
def open_file():
    global x; global sr
    filename = askopenfilename(
        filetypes = [ ("OGG Files", "*.ogg"), ("MP3 Files", "*.mp3"), ("WAV Files", "*.wav")]

        #onset
        hop_length = 100
        onset_env = librosa.onset.onset_strength(y=x, sr=sr, hop_length=hop_length)
        onset_samples = librosa.onset.onset_detect(y=x,
                                                    sr=sr, units='samples',
                                                    hop_length=hop_length,
                                                    backtrack=False,
                                                    pre_max=20,
                                                    post_max=20,
                                                    pre_avg=100,
                                                    post_avg=100,
                                                    delta=0.2,
                                                    wait=0)
        onset_boundaries = numpy.concatenate([[0], onset_samples, [len(x)]])

        onset_times = librosa.samples_to_time(onset_boundaries, sr=sr)
        librosa.display.waveshow(y=x, sr=sr)
        plt.vlines(onset_times, -1, 1, color='r')
        plt.savefig("onset.png")#onset *****
        plt.clf()
    if
        return
    plt.rcParams['figure.figsize'] = (22, 10.5)
    bins_per_octave = 36
    #something here
```

format="wav")

Outputs the waveform of the audio file and marks the onset frames with red vertical lines



```
#定義按鈕指令
def open_file():
    global x; global sr
    #something here

#cqt
cqt = librosa.cqt(x, sr = sr, n_bins = 300, bins_per_octave = bins_per_octave)
log_cqt = librosa.amplitude_to_db(numpy.abs(cqt))
spec = librosa.display.specshow(log_cqt, sr = sr, x_axis = "time", y_axis = "cqt_note",
                                bins_per_octave = bins_per_octave)
plt.grid(which = "both",color='g',axis='y',linewidth=0.7)
plt.savefig("spec_cqt.png")#cqt *****
plt.grid(which = "both",color='w',axis='y',linewidth=0.7)
plt.savefig("spec_cqt_temp.png")#cqt for bm
image = Image.open("spec_cqt_temp.png")
gray = image.convert("L")
threshold = 150
image_threshold = gray.point(lambda x: 255 if x > threshold else 0)
image_threshold.save("bm_spec_cqt.png")#cqt with bm *****
plt.grid(which = "both",color='g',axis='y',linewidth=0.7)
for i in onset_times:
    plt.axvline(i, -1, 1, color='w',lw=0.7)
plt.savefig("spec_cqt_tempo.png")#cqt with tempo *****
plt.grid(which = "both",color='w',axis='y',linewidth=0.7)
plt.savefig("spec_cqt_temp_tempo.png")#cqt with tempo for bm
image = Image.open("spec_cqt_temp_tempo.png")
gray = image.convert("L")
threshold = 150
image_threshold = gray.point(lambda x: 255 if x > threshold else 0)
image_threshold.save("bm_spec_cqt_tempo.png")#cqt with tempo,bm
*****
plt.clf()

#something here
```

#DEF ANALYZE

("WAV Files", "*.wav")]

".wav", format="wav")
av"

Generate the CQT graph
of the audio file



#DEF ANALYZE

```
#定義按鈕指令
def open_file():
    global x; global sr
    filename = askopenfilename(
        filetypes = [ ("OGG Files", "*.ogg"), ("MP3 Files", "*.mp3"), ("WAV Files", "*.wav") ]
    )
    if not filename:
        return

    #chroma_stft
    S = numpy.abs(librosa.stft(x))
    chroma = librosa.feature.chroma_stft(S=S, sr=sr)
    img = librosa.display.specshow(chroma, y_axis='chroma', x_axis='time')
    plt.grid(which = "both",color='g',axis='y',linewidth=0.7)
    plt.savefig("spec_chromaSTFT.png")#chromaSTFT *****
    image = Image.open("spec_chromaSTFT.png")
    gray = image.convert("L")
    threshold = 150
    image_threshold = gray.point(lambda x: 255 if x > threshold else 0)
    image_threshold.save("bm_spec_chromaSTFT.png")#chromaSTFT with bm
    *****
    for i in onset_times:
        plt.axvline(i, -1, 1, color='w',lw=0.7)
    plt.savefig("spec_chromaSTFT_tempo.png")#chromaSTFT with tempo *****
    plt.clf()

    return

plt.rcParams['figure.figsize'] = (22, 10.5)
bins_per_octave = 36
#something here
```

format="wav")

Generate the STFT graph of the audio file



```
#定義按鈕指令
def open_file():
    global x; global sr
    filename = askopenfilename(
        filetypes = [ ("OGG Files", "*.ogg"), ("MP3 Files", "*.mp3"), ("WAV Files", "*.wav") ]
    )
    if not filename:

        #final
        image = Image.open("spec_chromaSTFT_tempo.png")
        gray = image.convert("L")
        threshold = 150
        image_threshold = gray.point(lambda x: 255 if x > threshold else 0)
        image_threshold.save("bm_spec_chromaSTFT_tempo.png")#chromaSTFT with tempo,bm
        ****
        cur_img = "bm_spec_chromaSTFT_tempo.png"
        spec_image = ImageTk.PhotoImage(Image.open("bm_spec_chromaSTFT_tempo.png"))
        panel.configure(image = spec_image)
        panel.image = spec_image
        plt.clf()
    def
        global x; global sr; global spec_image; global cur_img
        cur_img="spec_chromaSTFT.png"
        if sr == 0:
            return
        plt.rcParams['figure.figsize'] = (22, 10.5)
        bins_per_octave = 36
        #something here
format="wav")
```

Generate the graphs with binary mask and tempo filters

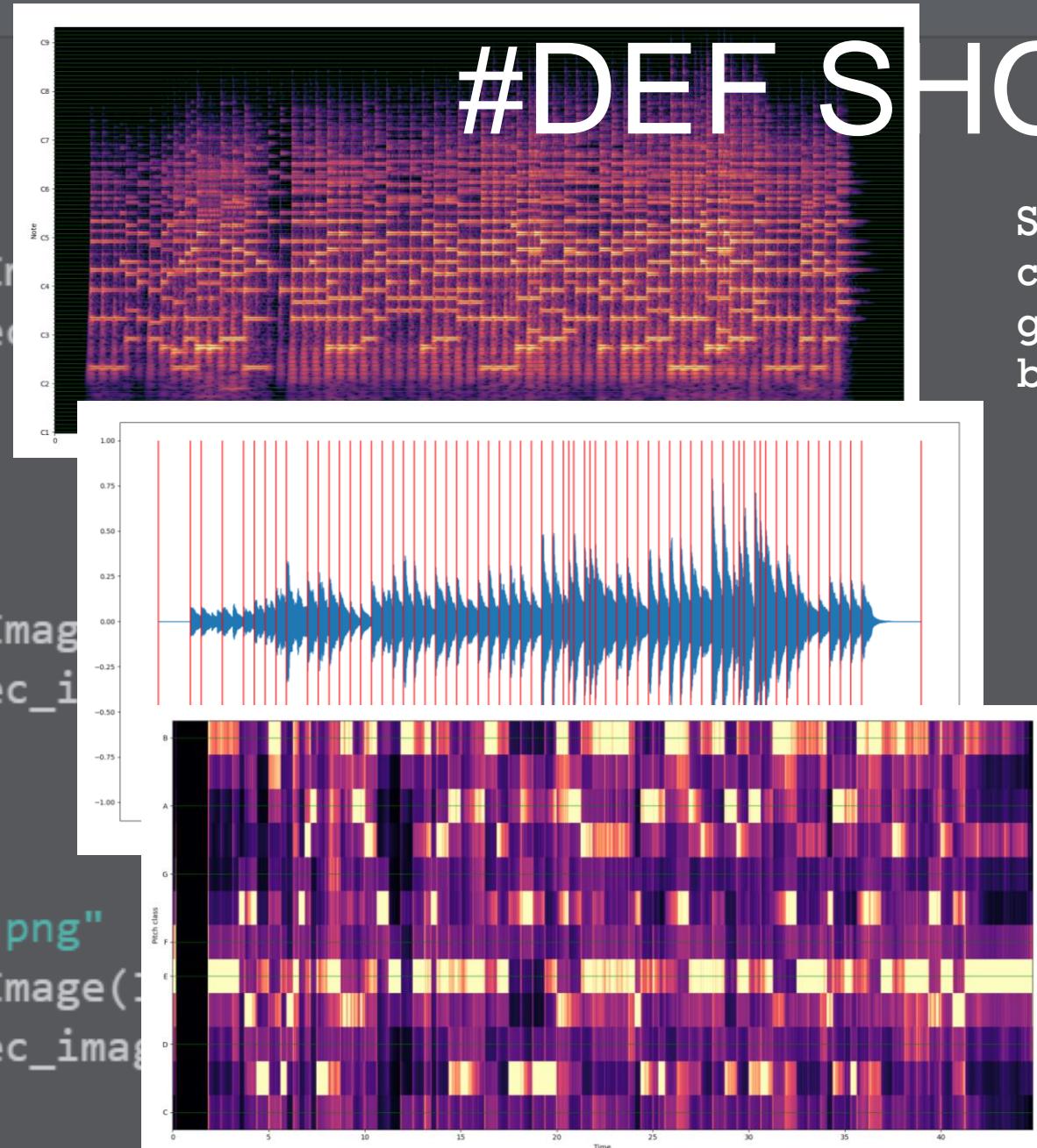




```
def show_cqt():
    global cur_img
    cur_img = "spec_cqt.png"
    spec_image = ImageTk.PhotoImage(file=cur_img)
    panel.configure(image = spec_image)
    panel.image = spec_image

def show_onset():
    global cur_img
    cur_img = "onset.png"
    spec_image = ImageTk.PhotoImage(file=cur_img)
    panel.configure(image = spec_image)
    panel.image = spec_image

def show_stft():
    global cur_img
    cur_img = "spec_chromaSTFT.png"
    spec_image = ImageTk.PhotoImage(file=cur_img)
    panel.configure(image = spec_image)
    panel.image = spec_image
```

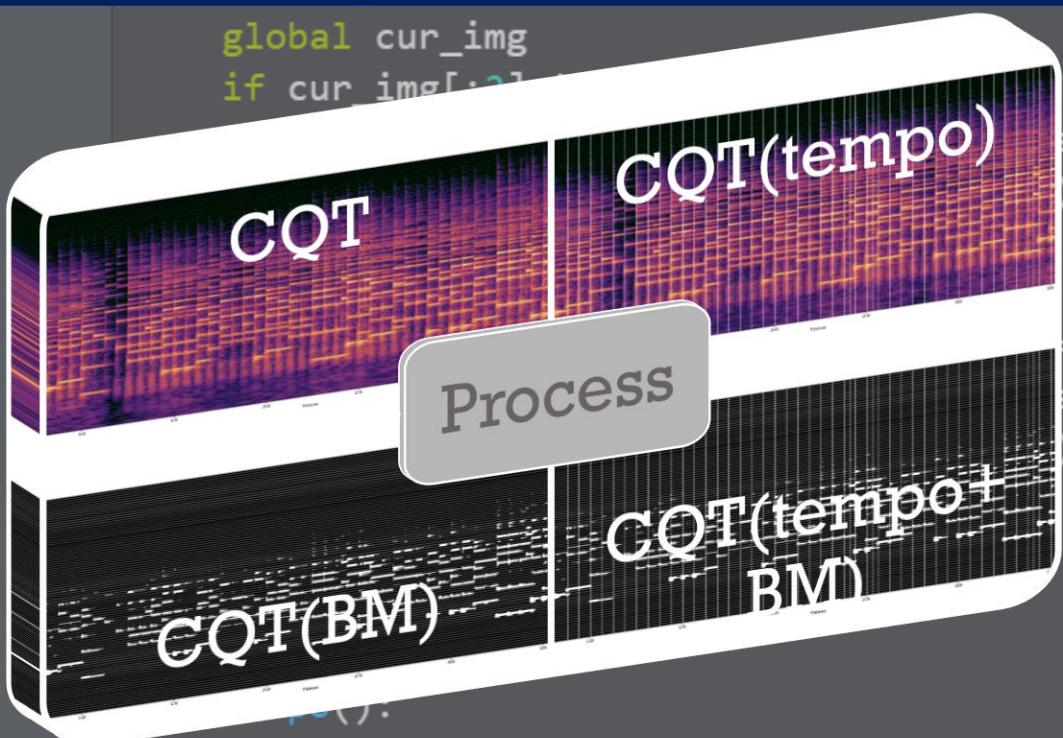


#DEF SHOW

Shows the corresponding graph of the button pressed



#DEF INFORMATION PROCESS



```
global cur_img
if cur_img[-1] != "onset.png":
    image.open("bin_" + cur_img)
else:
    cur_img = "onset.png"
    image.open(cur_img[3:])

#DEF INFORMATION PROCESS

#DEF CQT(tempo+BM)

#DEF CQT(tempo)

#DEF CQT
```

```
global cur_img
if cur_img[-9:-4] != "tempo" and cur_img != "onset.png":
    spec_image = ImageTk.PhotoImage(Image.open(cur_img[:-4]+"_tempo.png"))
    panel.configure(image = spec_image)
    panel.image = spec_image
    cur_img = cur_img[:-4]+"_tempo.png"
elif cur_img[-9:-4] == "tempo" and cur_img != "onset.png":
    spec_image = ImageTk.PhotoImage(Image.open(cur_img[:-10]+".png"))
    panel.configure(image = spec_image)
    panel.image = spec_image
    cur_img = cur_img[:-10] + ".png"
else:
    pass
```

Making the tempo filter

驚濤心電圖 今天 #3:56 GUI INITIALIZATION



#初始設定

```
window = tk.Tk()
window.title("MusicTranscription")
window.geometry("1920x1080")
x = []
sr = 0
spec_image = None
window.columnconfigure(0, weight = 1)
window.rowconfigure(2, weight = 1)
```

Initialization of variables and
GUI widgets





驚濤心電圖 今天 13:56



#GUI BUTTONS

```
#按鈕
frm_buttons = tk.Frame(window, relief = "groove", bd = 2)
frm_buttons.grid(row = 0, column = 0, sticky = "ew")
btn_open = tk.Button(frm_buttons, text = "Open", command = open_file)
btn_open.grid(row = 0, column = 0, sticky = "ew", pady = 5, padx = 5)
btn_analyze = tk.Button(frm_buttons, text = "Analyze", command = analyze)
btn_analyze.grid(row = 0, column = 1, sticky = "ew", pady = 5, padx = 5)
btn_cqt = tk.Button(frm_buttons, text = "CQT", command = show_cqt)
btn_cqt.grid(row = 0, column = 2, sticky = "ew", pady = 5)
btn_onset = tk.Button(frm_buttons, text = "onset", command = show_onset)
btn_onset.grid(row = 0, column = 3, sticky = "ew", pady = 5)
btn_stft = tk.Button(frm_buttons, text = "Chroma ftst", command = show_stft)
btn_stft.grid(row = 0, column = 4, sticky = "ew", pady = 5)
btn_tempo = tk.Button(frm_buttons, text = "tempo", command = tempo)
btn_tempo.grid(row = 0, column = 5, sticky = "ew", pady = 5, padx = 5)
btn_bm = tk.Button(frm_buttons, text = "binary mask", command = binary_mask)
btn_bm.grid(row = 0, column = 6, sticky = "ew", pady = 5)
btn_close = tk.Button(frm_buttons, text = "Exit", command = quit)
btn_close.grid(row = 0, column = 7, sticky = "ew", pady = 5, padx = 20)
```

Music Transcription

Open Analyze CQT onset Chroma ftst tempo binary mask Exit
empty

Initialization of GUI
buttons



工作分配

張程翔：概念發想和音樂理論研究

汪晁安：Librosa 分析

耿睿棋：GUI 實作

