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

Music Transcription

Music Processing

Application for Python

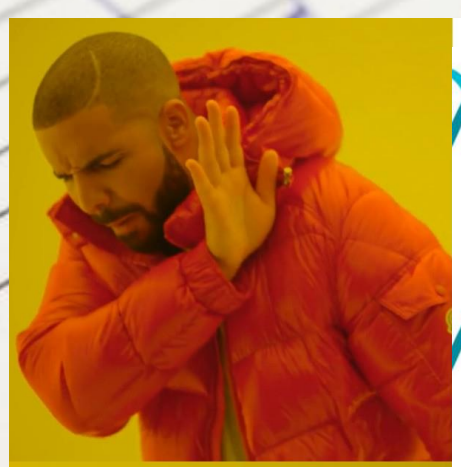
PROJECT IDEA

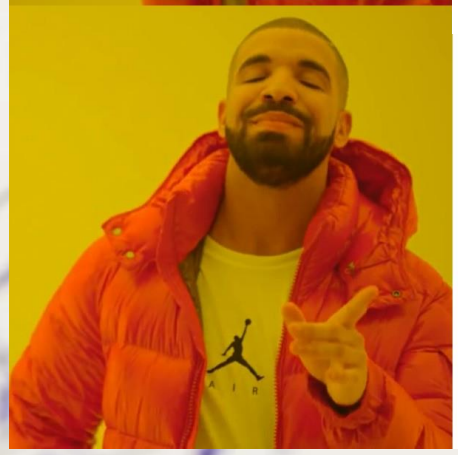
- Listen to music
- Wonder how to play it
- Using Python as a tool

librosa

MUSIC TRANSCRIPTION

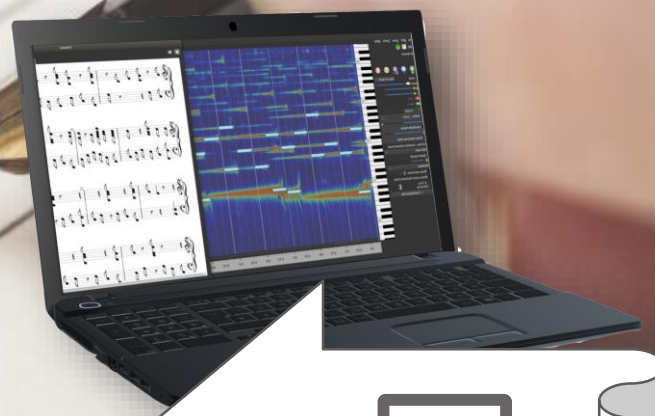
What is transcription?

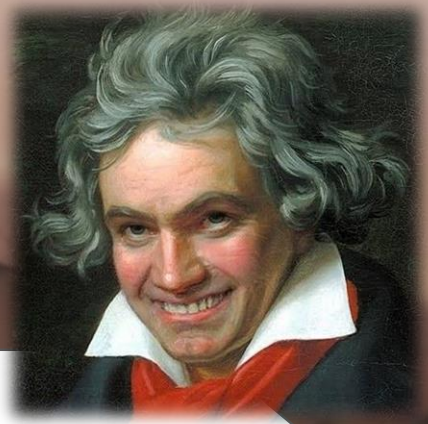




採譜（transcription）亦稱轉譜，指的是透過人耳對一首樂曲的反覆試聽，從而將其總譜寫下來的過程。

How to transcript music?





MUSIC PROCESSING

A frontiers for us to explore

High-level mathematics applied

Music theory? What's that?

HIGH-LEVEL MATHEMATICS?

Don't forget some matri

$$\mathrm{DFT}_N \cdot \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{pmatrix} = \left(\begin{array}{c|c} \mathrm{id}_M & \Delta_M \\ \hline \mathrm{id}_M & -\Delta_M \end{array}\right) \left(\mathrm{DF}$$

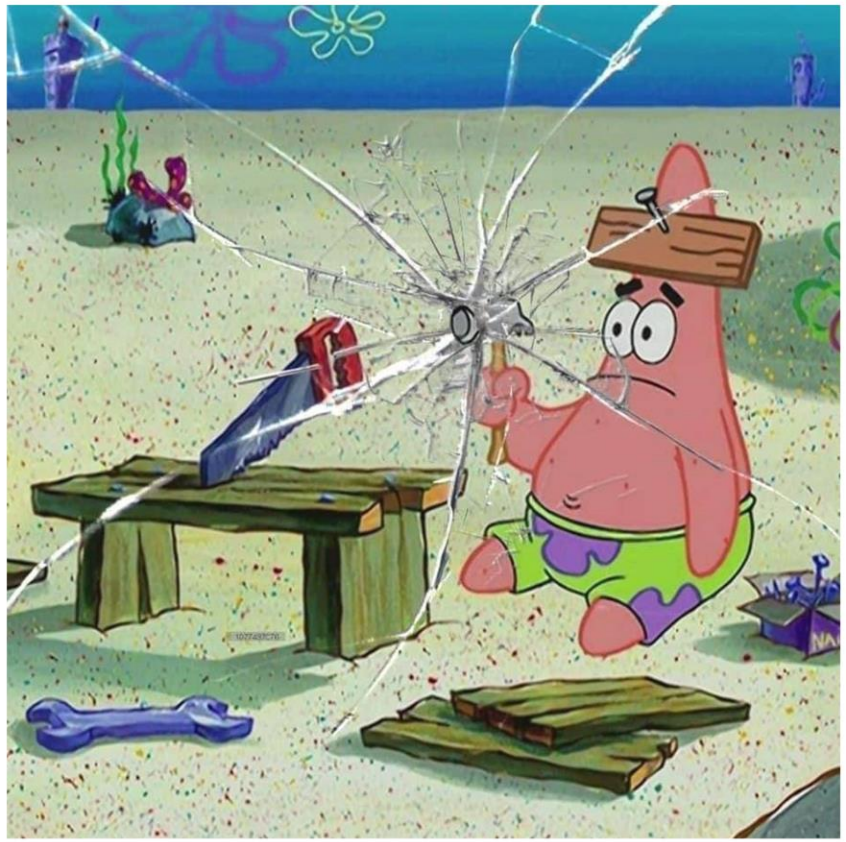
Fou

Based on the similarity measure (2.3), we compare t
soids $g = \mathbf{cos}_{\omega,\varphi}$ as defined in (2.2). For a fixed frequ

$$d_\omega := \max_{\varphi\in[0,1)} \left(\int_{t\in\mathbb{R}} f(t)\mathbf{cos}_{\omega,\varphi}($$

$$\varphi_\omega := \operatorname*{argmax}_{\varphi\in[0,1)} \left(\int_{t\in\mathbb{R}} f(t)\mathbf{cos}_{\omega,\varphi}$$

As previously discussed, the magnitude coefficient
frequency ω within the signal f. Additionally, the phase coefficient $\varphi_\omega \in [0,1$



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

$$\exp(-2\pi i\omega nT) \approx \hat{f}(\omega). \tag{2.21}$$

Riemann sum

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

np into the state α_{i_1} (with probability c_{i_1}), then it
noves on to state α_{i_2} (with probability $a_{i_1 i_2}$), and

'bout some probabilities?

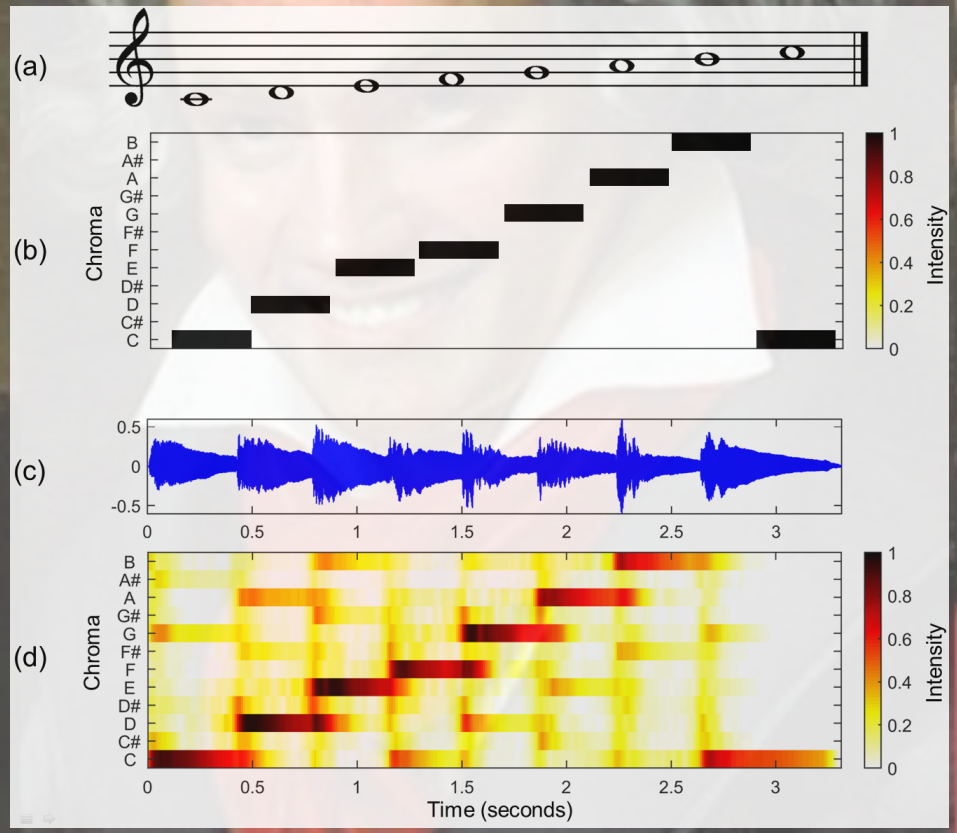
$$_1k_1 \cdot a_{i_1 i_2} \cdot b_{i_2 k_2} \cdot \ldots \cdot a_{i_{N-1} i_N} \cdot b_{i_N k_N}. \tag{5.32}$$

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

$$[O,S \mid \Theta] \tag{5.33}$$

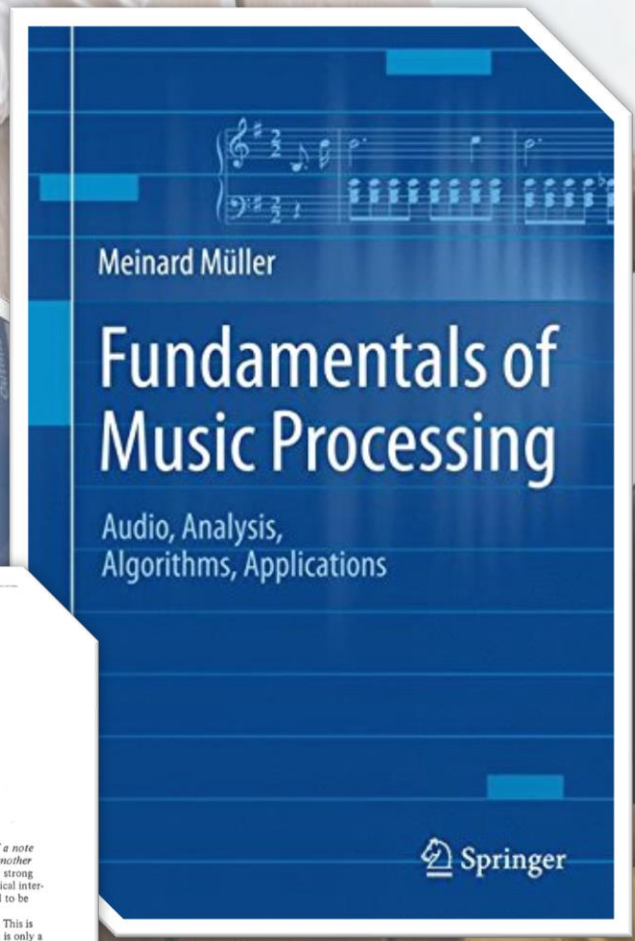
$$= \sum_{i_1=1} \sum_{i_2=1} \cdots \sum_{i_N=1} c_{i_1} \cdot b_{i_1 k_1} \cdot a_{i_1 i_2} \cdot b_{i_2 k_2} \cdot \ldots \cdot a_{i_{N-1} i_N} \cdot b_{i_N k_N}. \tag{5.34}$$

MUSIC THEORY



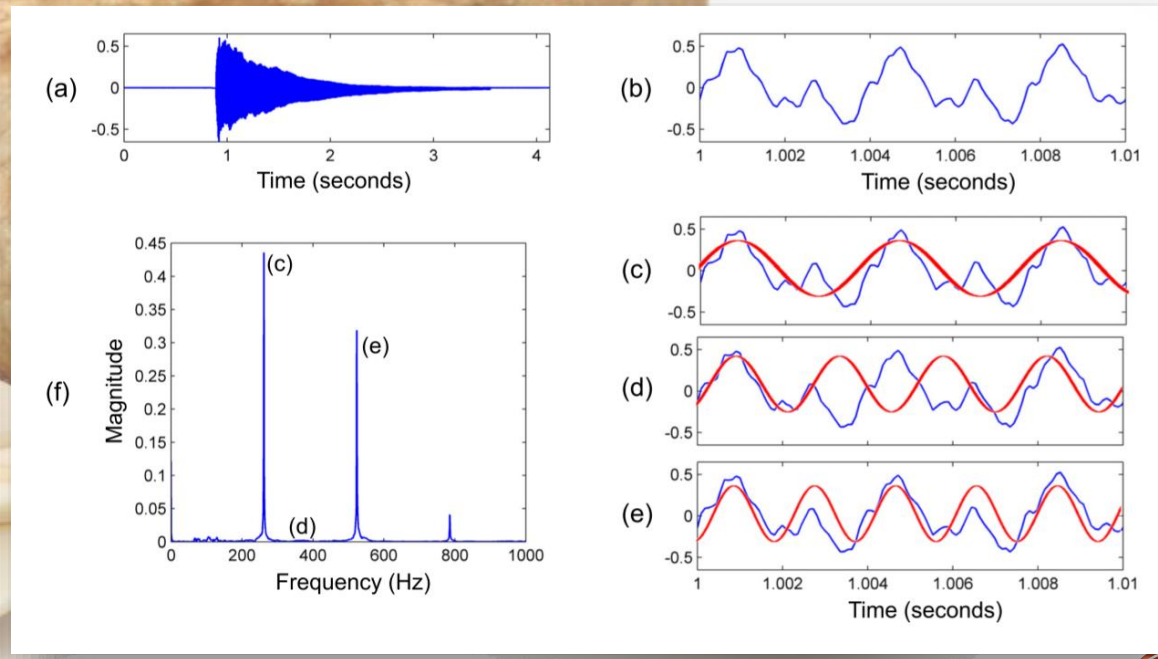
HOW DO WE OVERCOME IT?

librosa



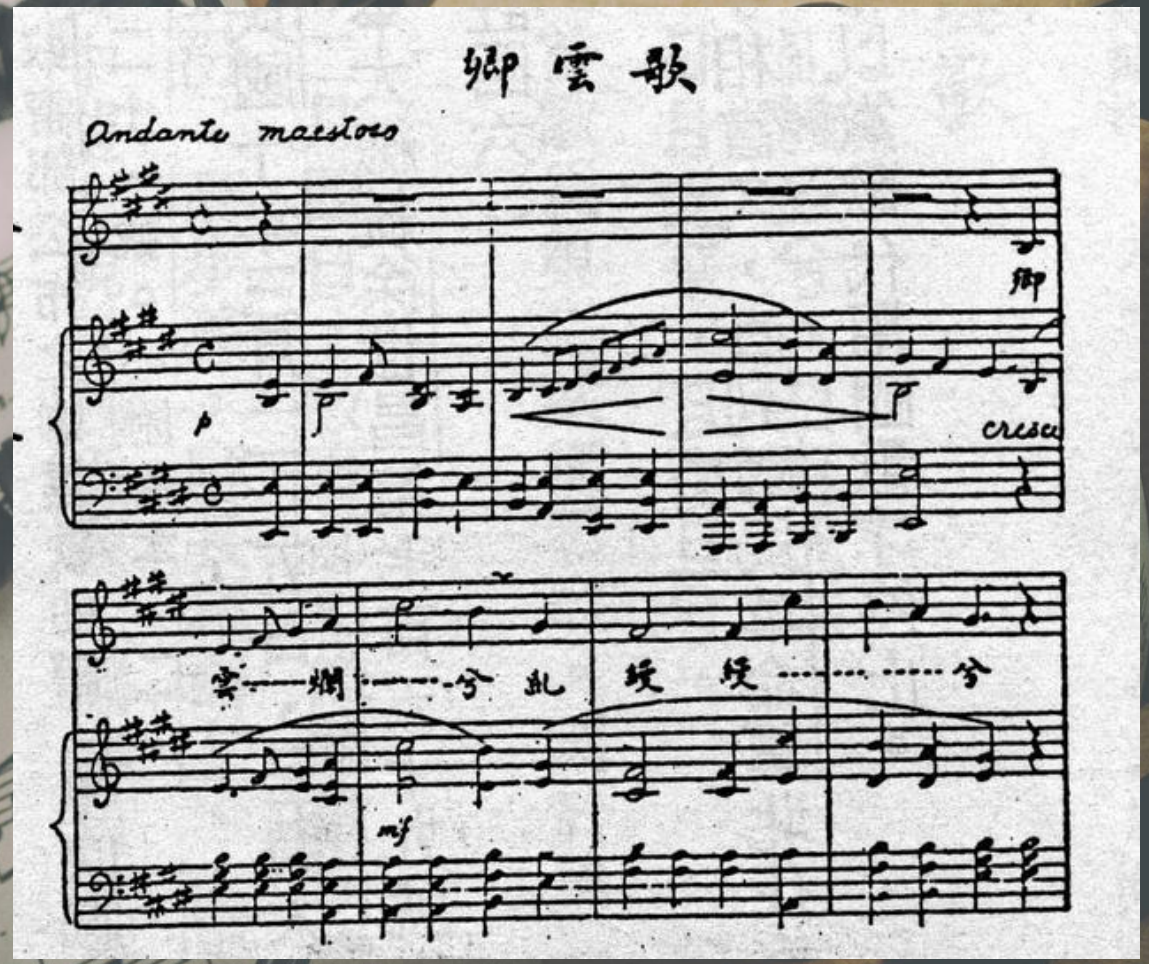
FT IN A PEANUT SHELL

$$F(\omega) = \mathcal{F}[f(t)] = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt$$

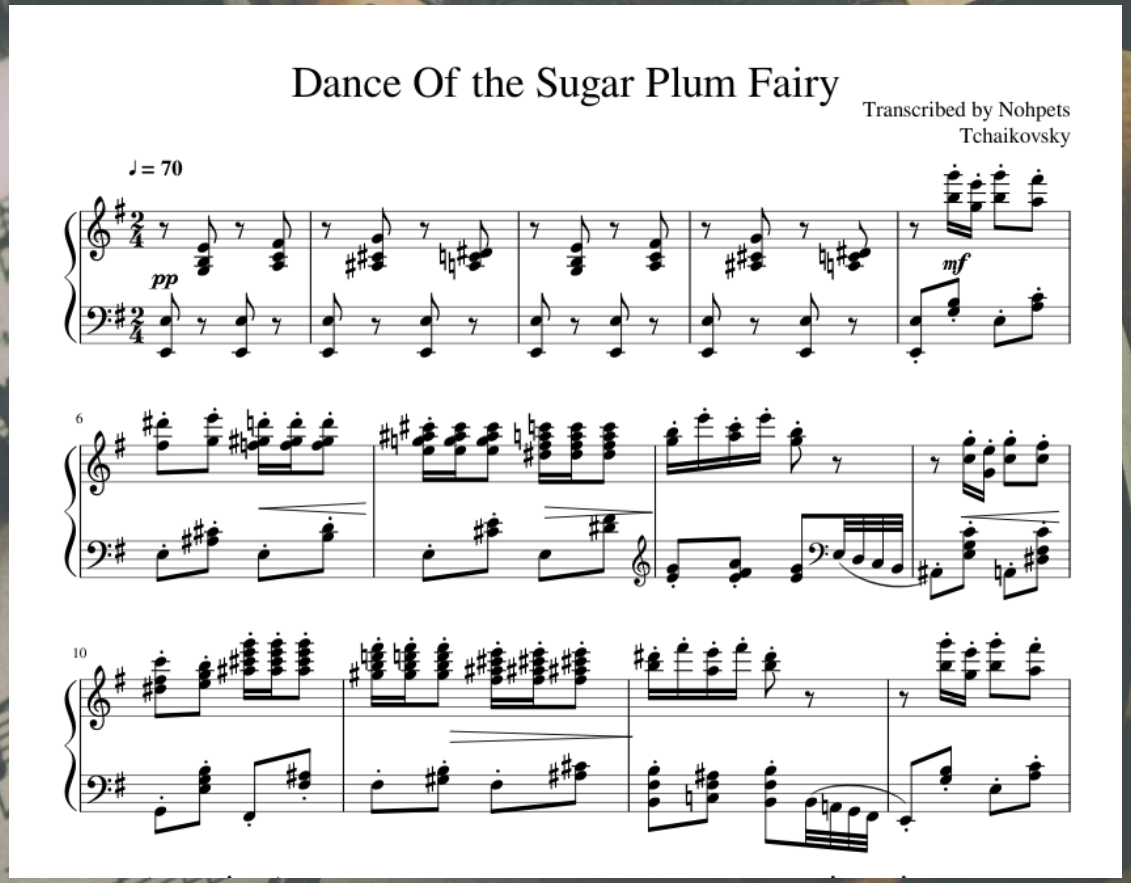


PRACTICAL APPLICATIONS

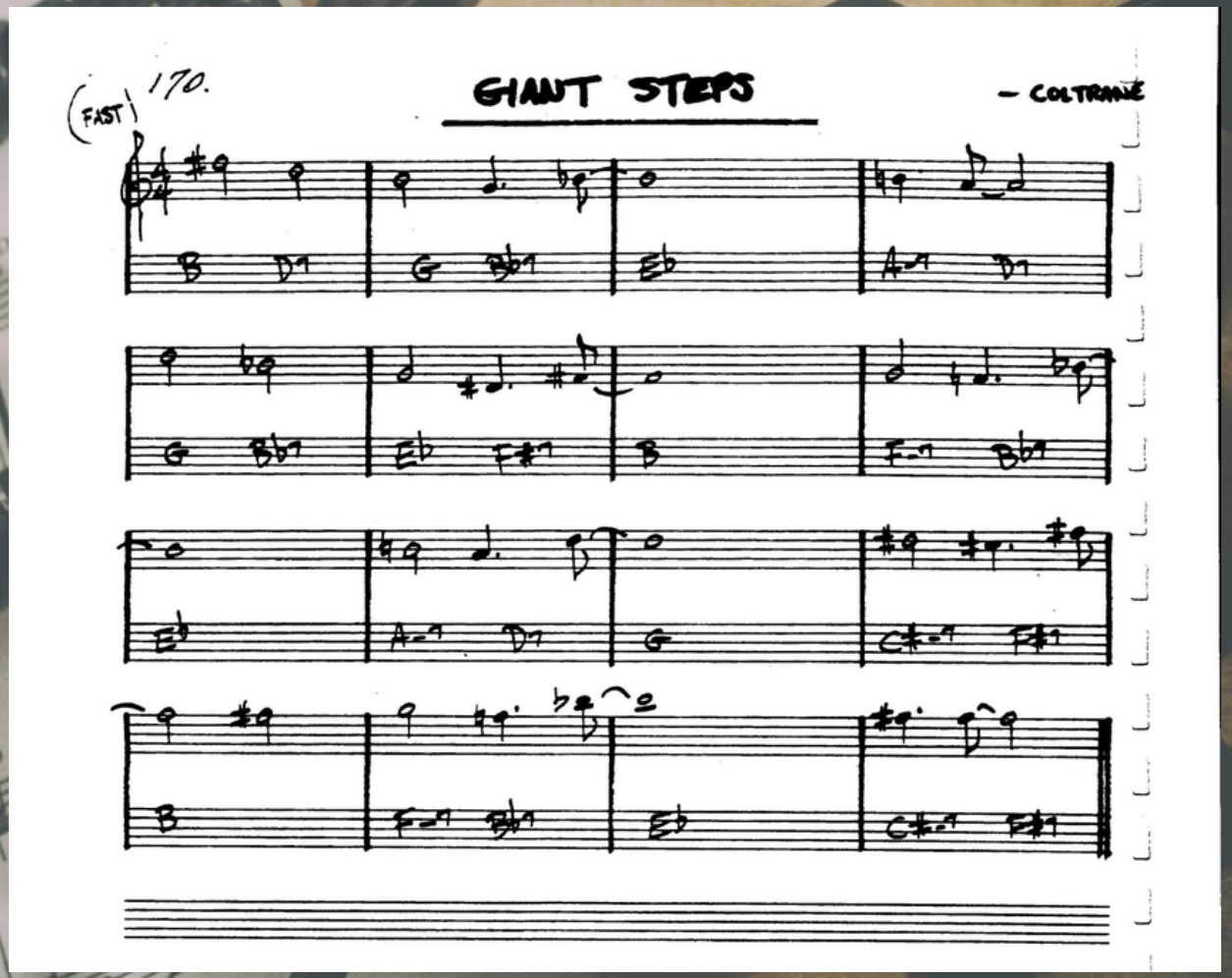
SAMPLE A



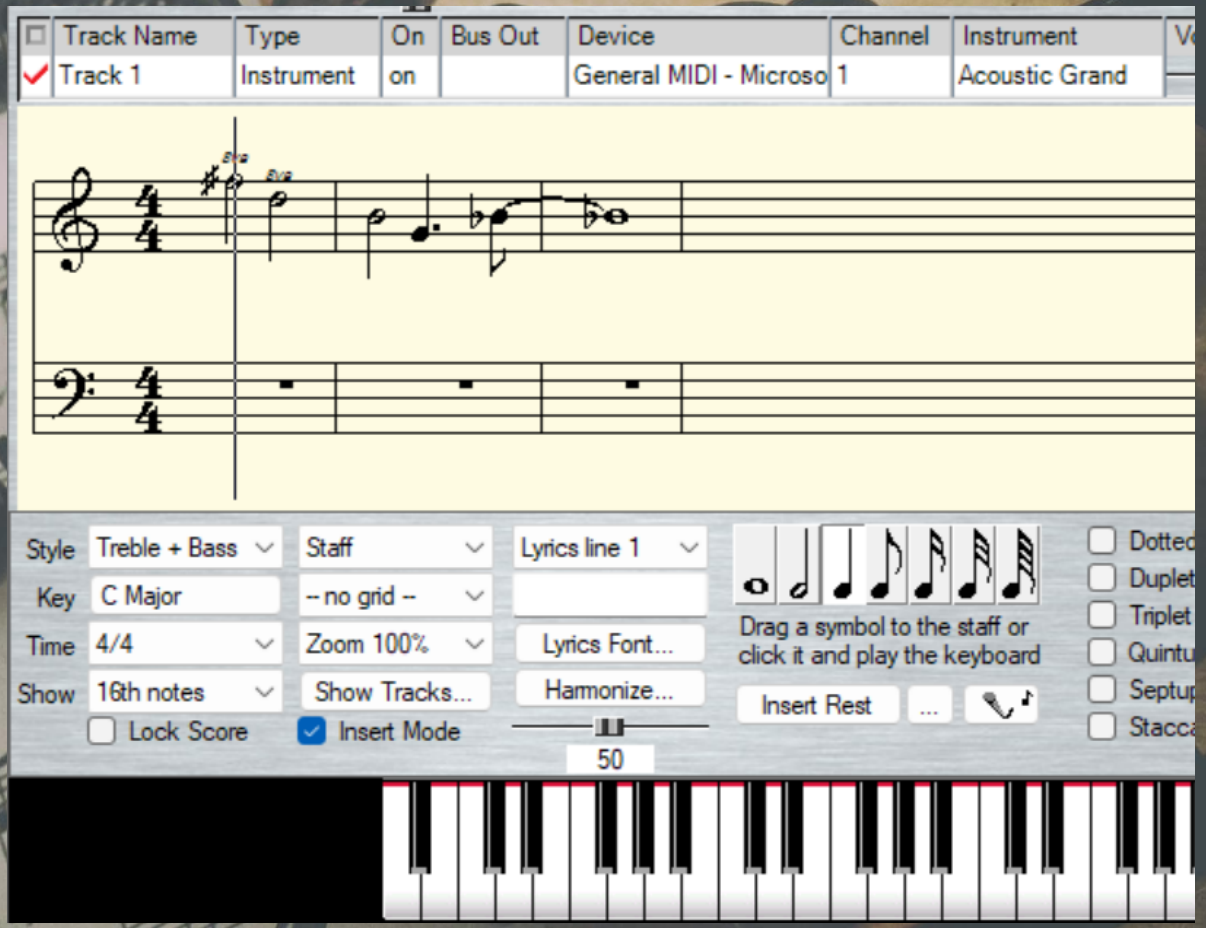
SAMPLE B


Dance Of the Sugar Plum Fairy
Transcribed by Nohpets
Tchaikovsky
♩= 70
pp
mf

ADDITIONAL-SAMPLE G



ADDITIONAL-SAMPLE G’



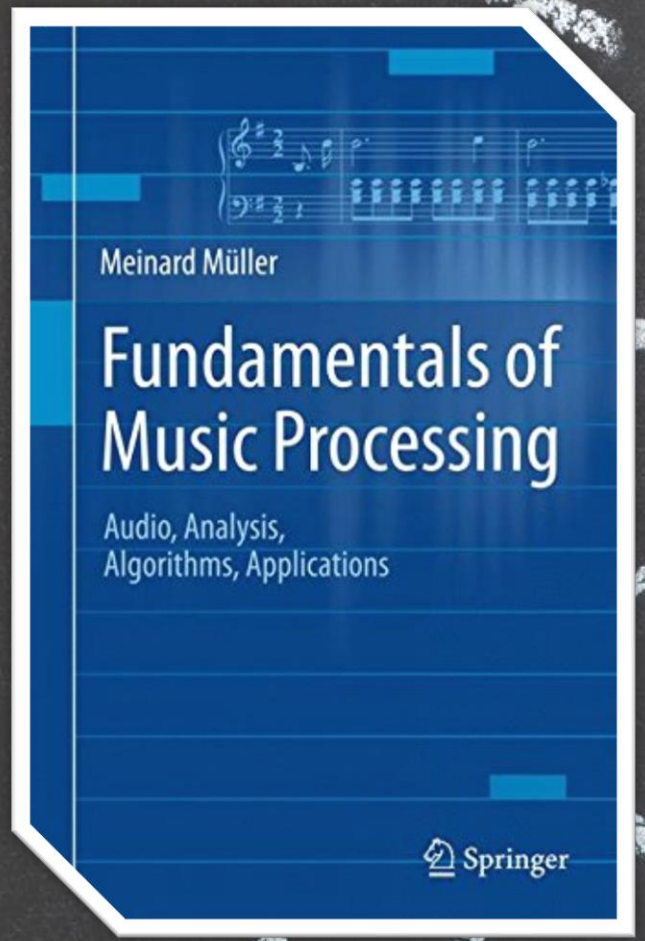
PROBLEMS WHEN ANALYZING

Too much information…

How to filter noise figure?

How to present?

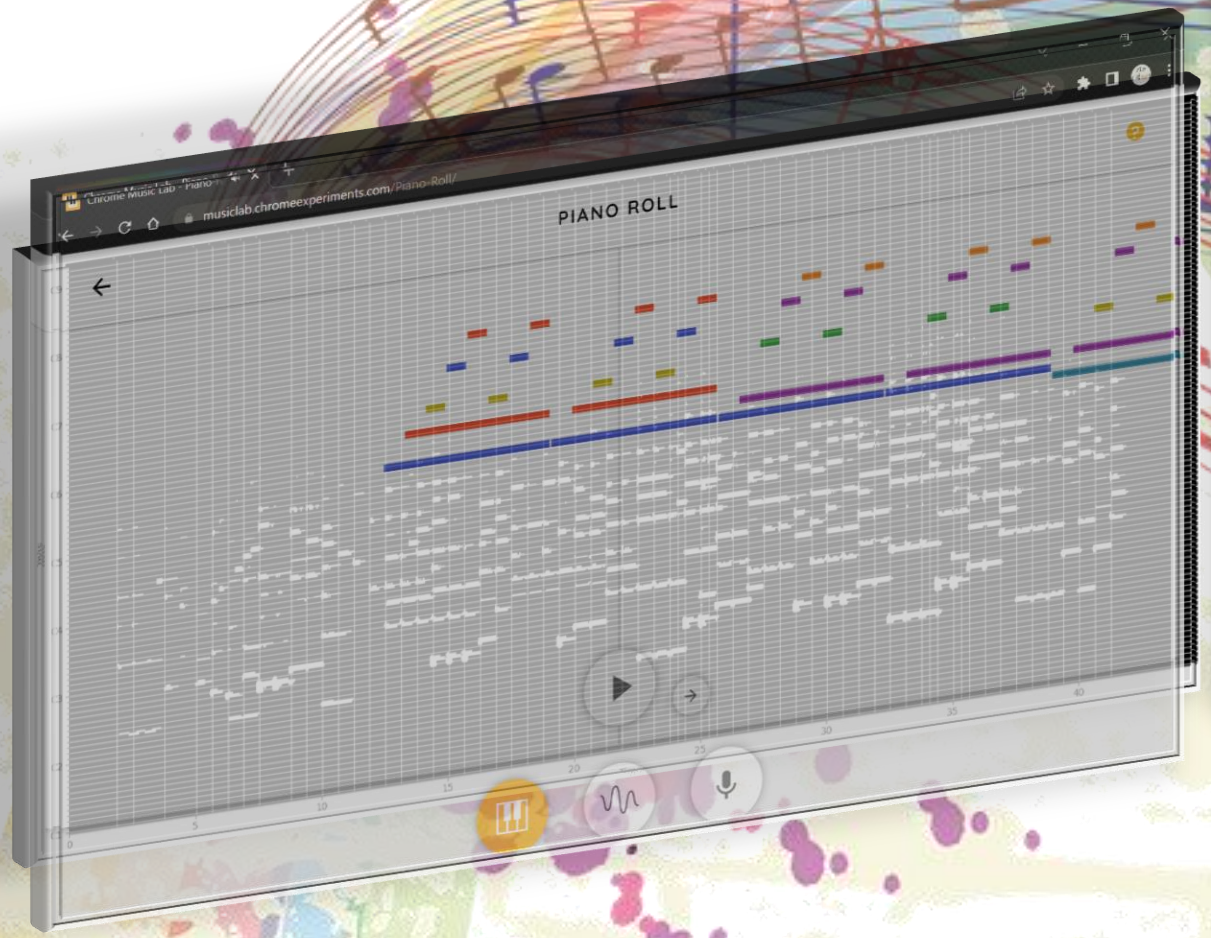
TOO MUCH INFORMATION

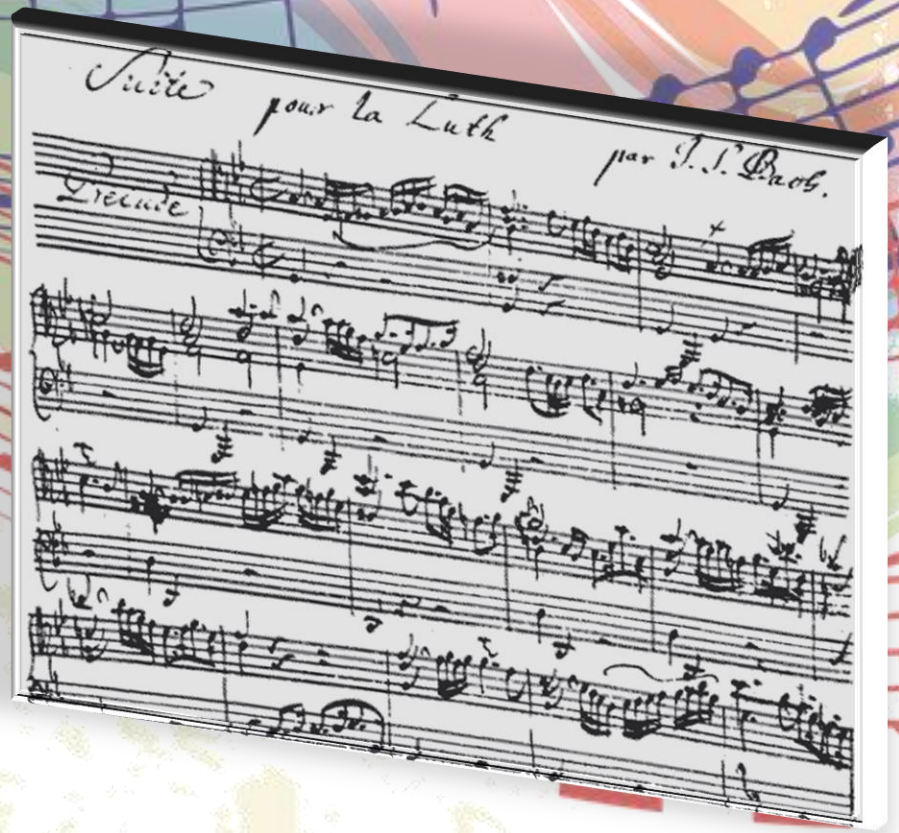


What does this project need?

Theorem? Math? HMM? MIDI?

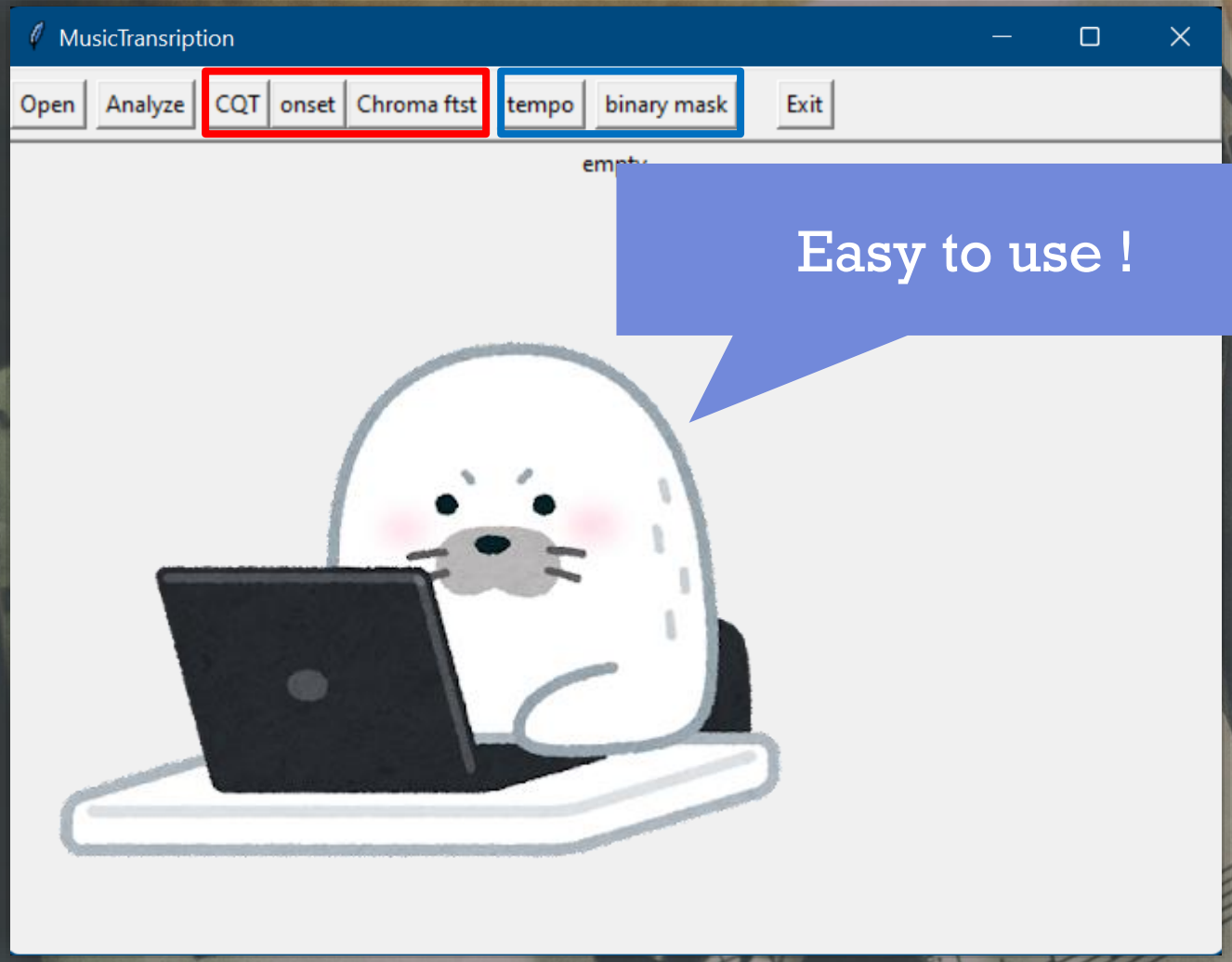
HOW TO PRESENT?





.MP3 VS .WAV

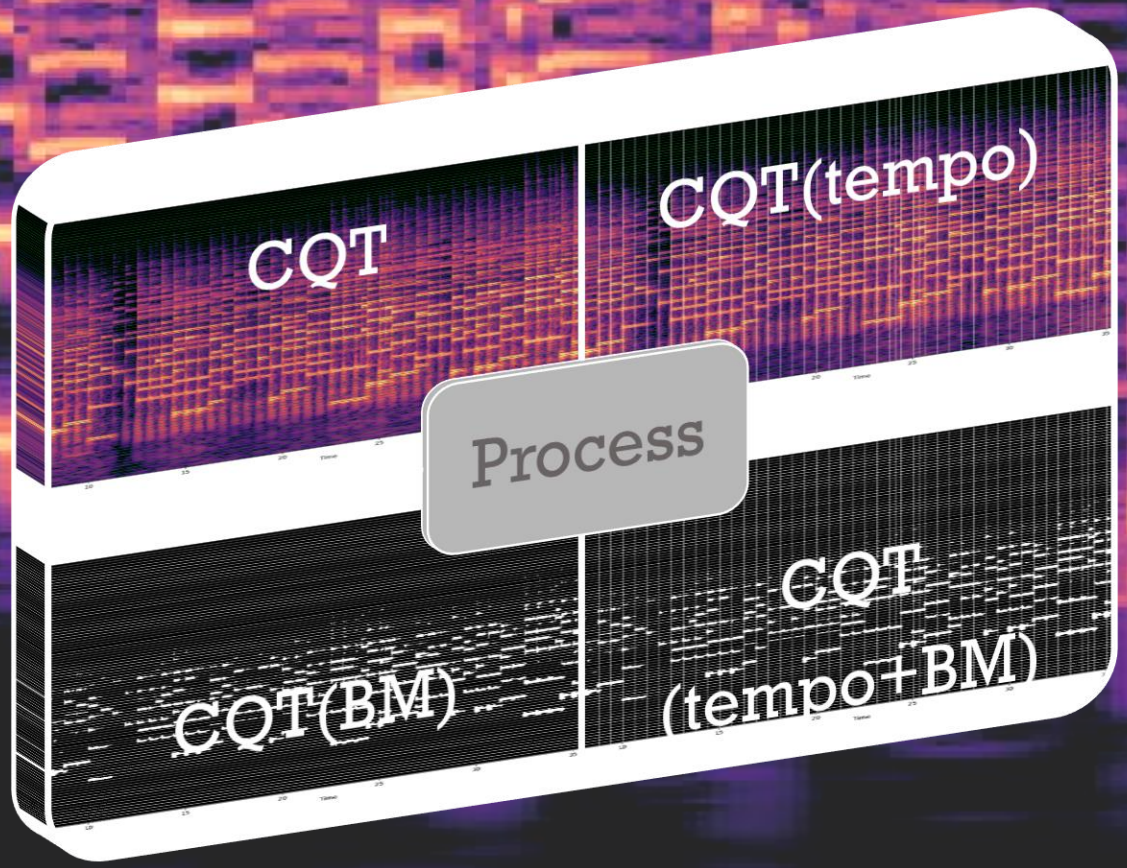
GUI

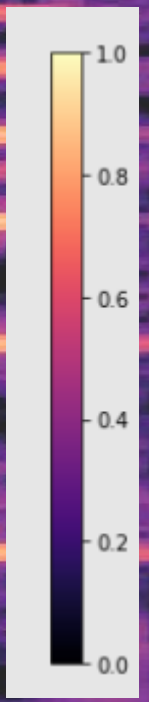


Presentation type

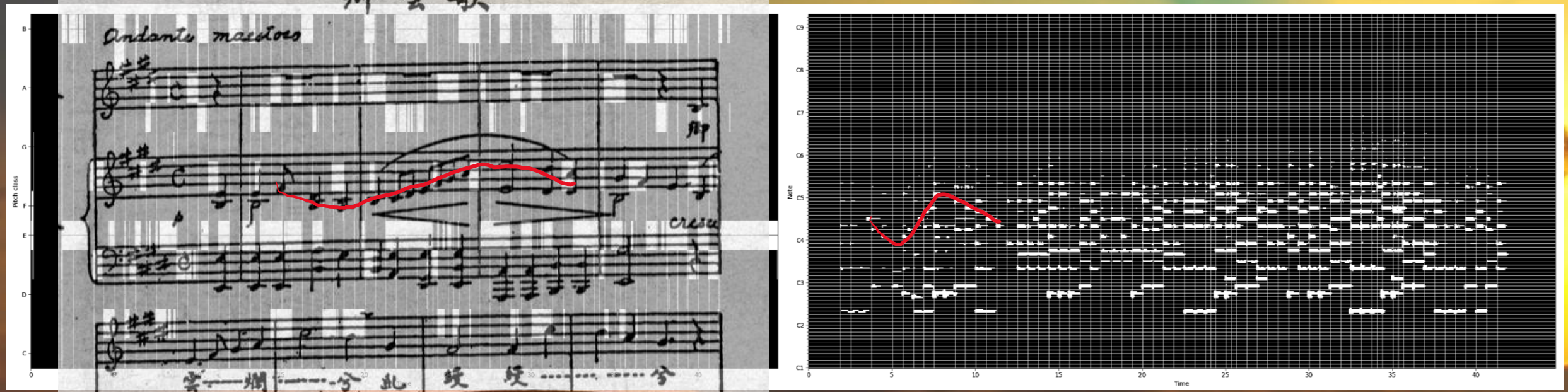
Onset_dictect & Filter

HOW TO FILTER NOISE FIGURE

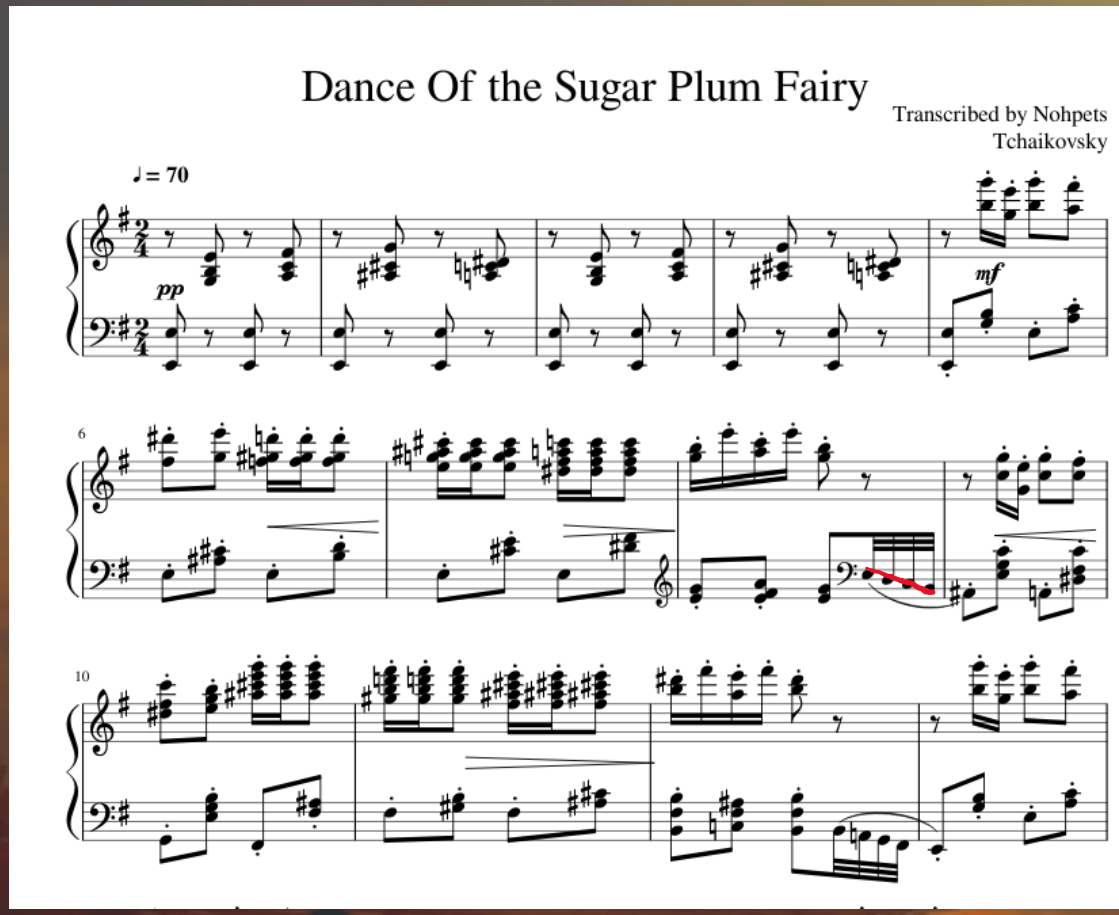


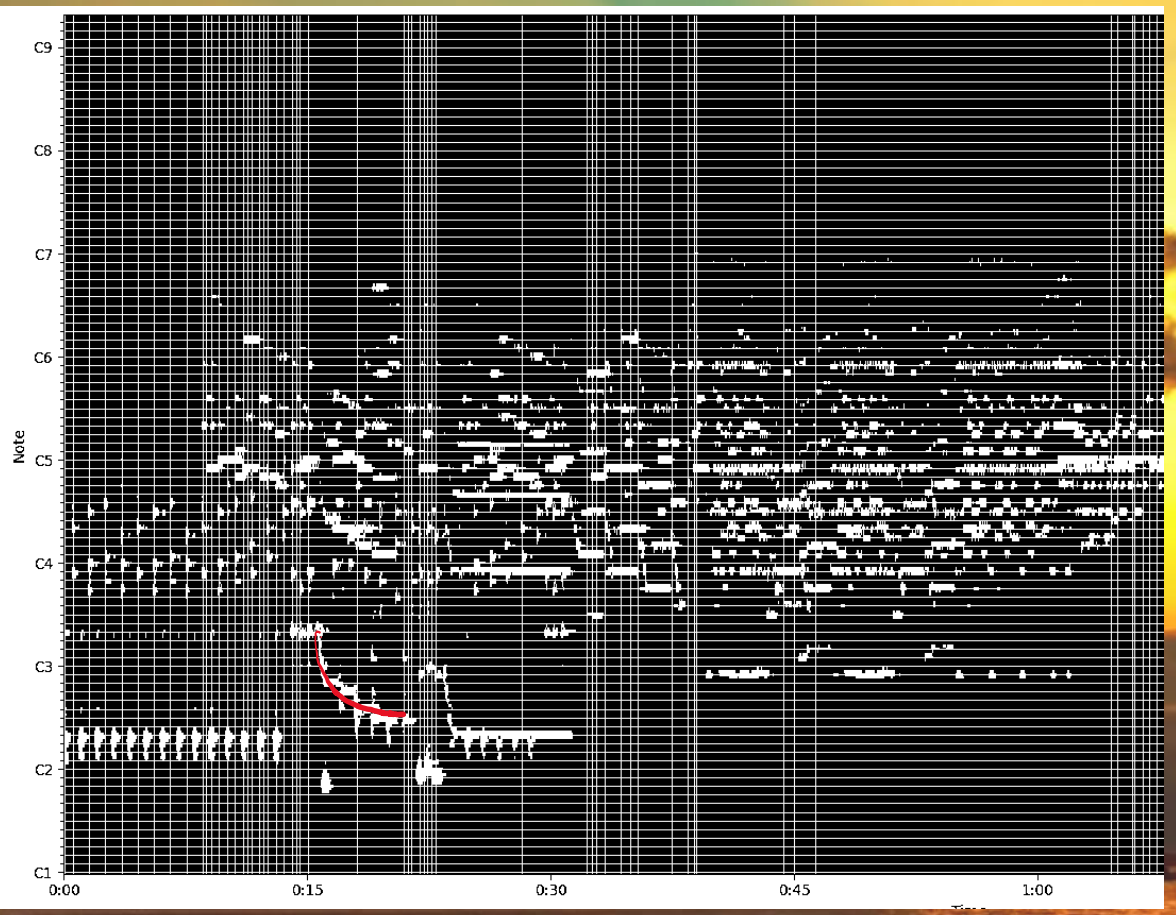


PERFORMANCE SAMPLE A



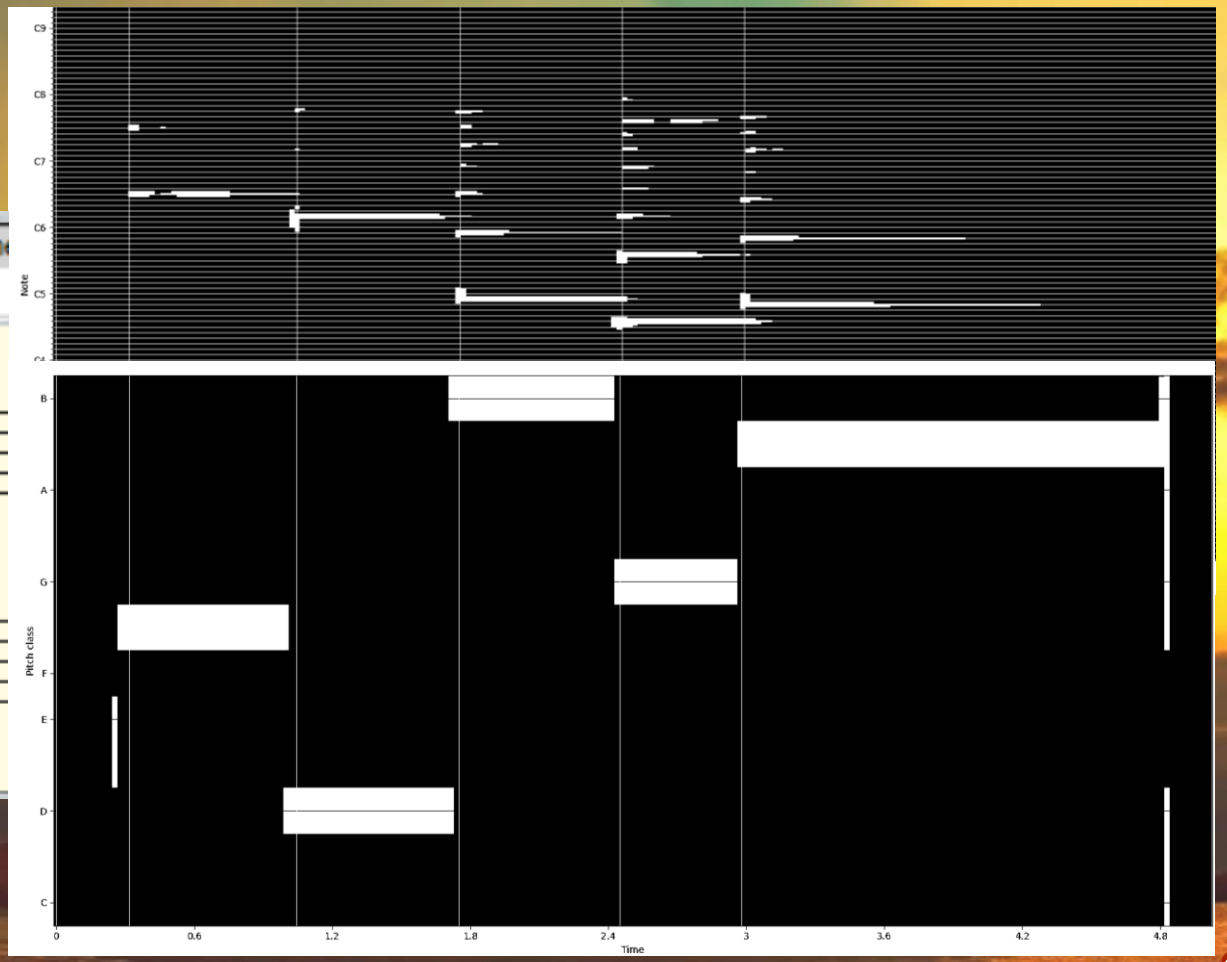
PERFORMANCE SAMPLE B



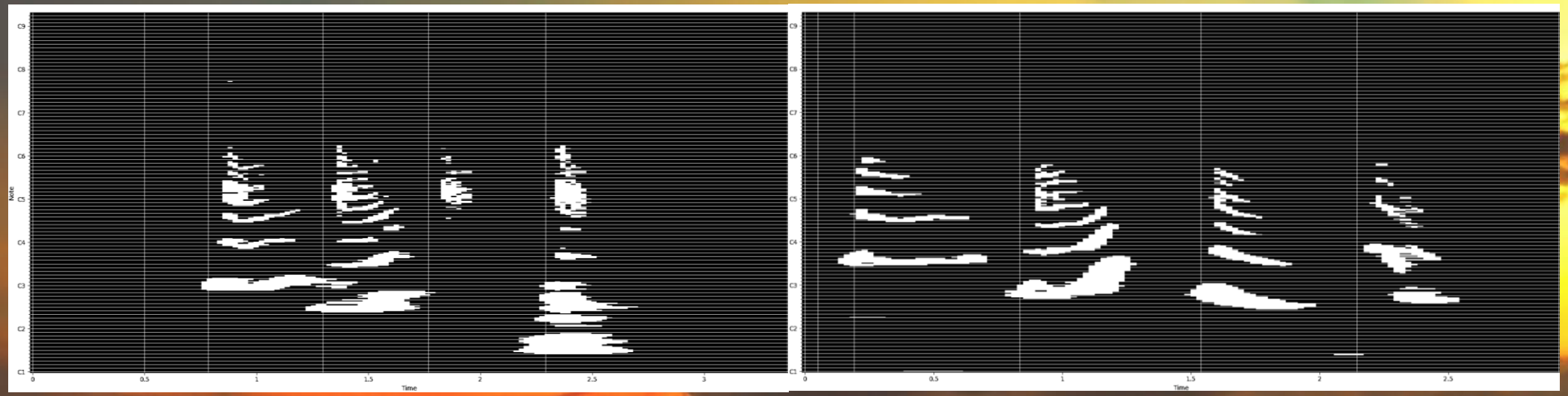


PERFORMANCE SAMPLE G’

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

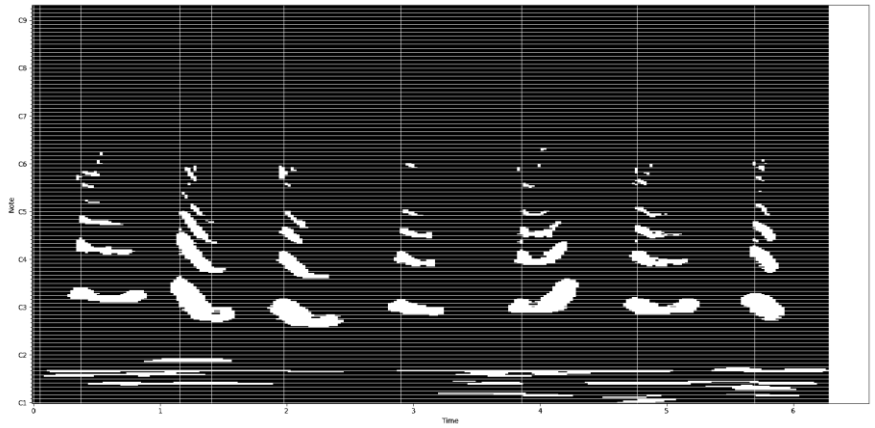


APPLICATION FOR TONES ANALYZING

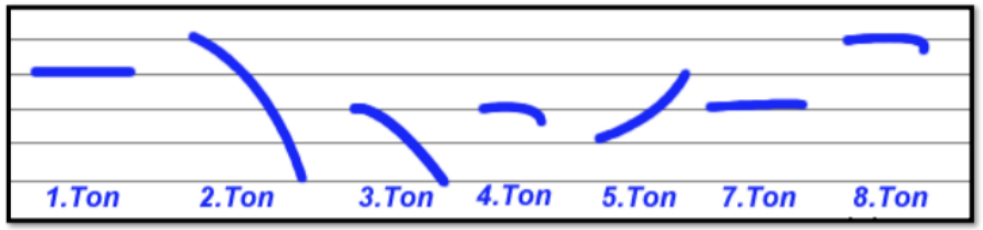


Different people say: “通同統痛”

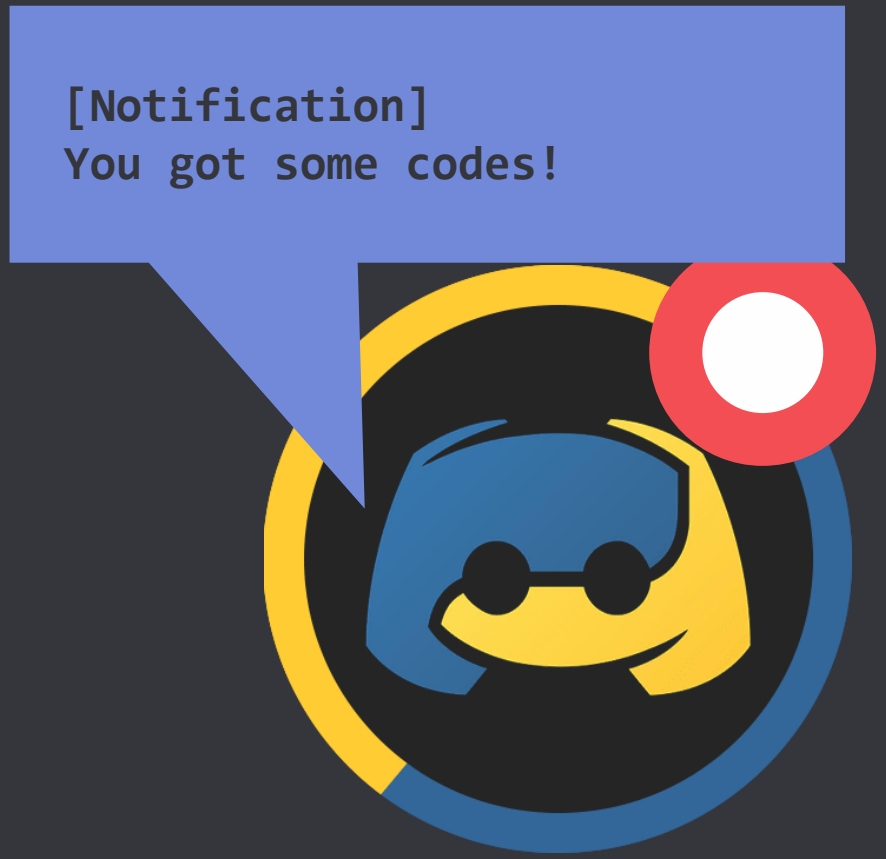
APPLICATION FOR TONES ANALYZING



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	˥	˩˧	˨˩	˥	˧		˨	˥

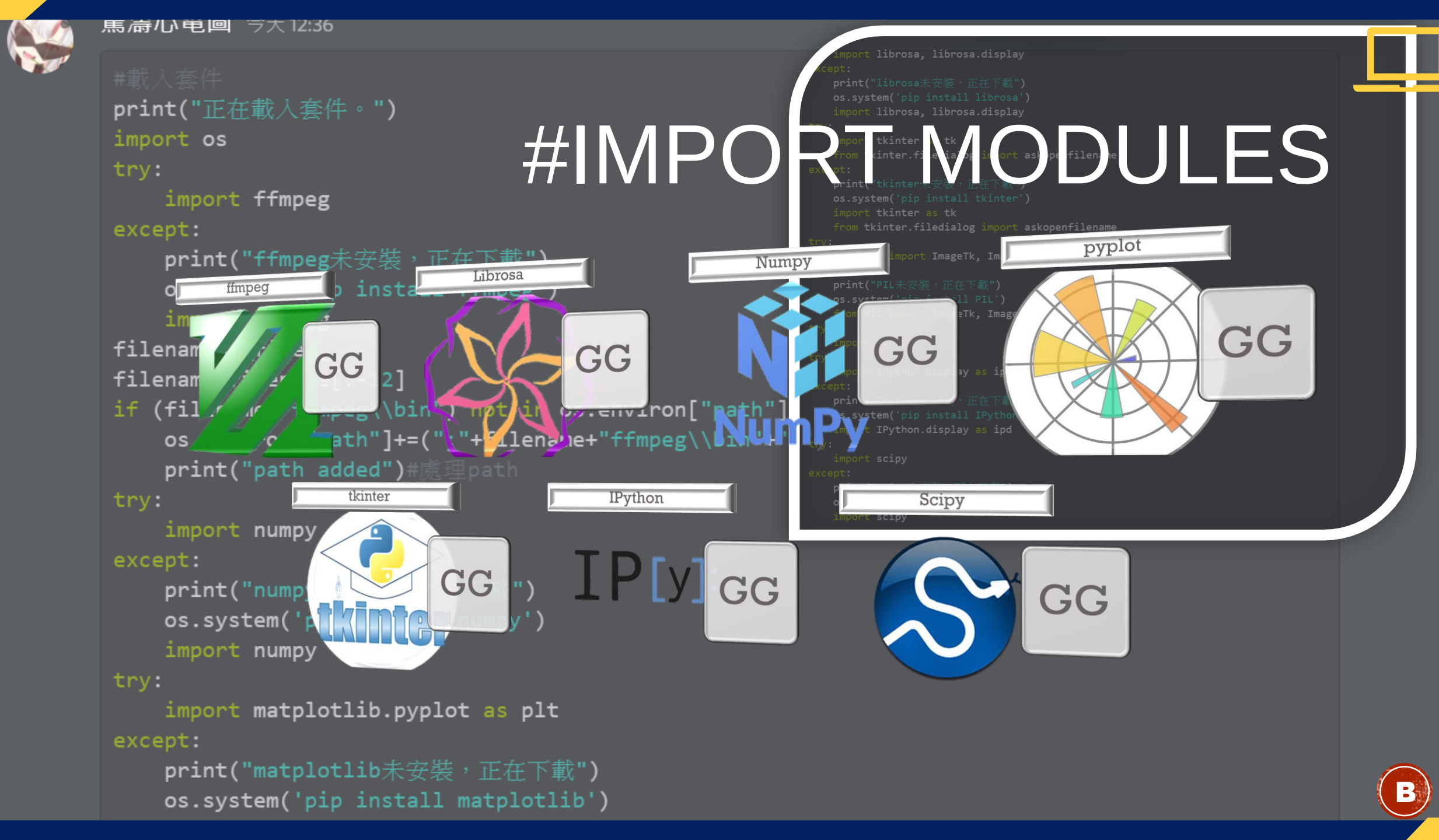


(右圖來自江文瑜教授對臺語音調所提供的參考資料)



Implementation

#IMPORT MODULES



#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
    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
```

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()
```

```
de
    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

#DEF ANALYZE

```
#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()
```

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

```
    #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()
```

```
format="wav")
```

Generate the STFT 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:
```

```
    #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()
```

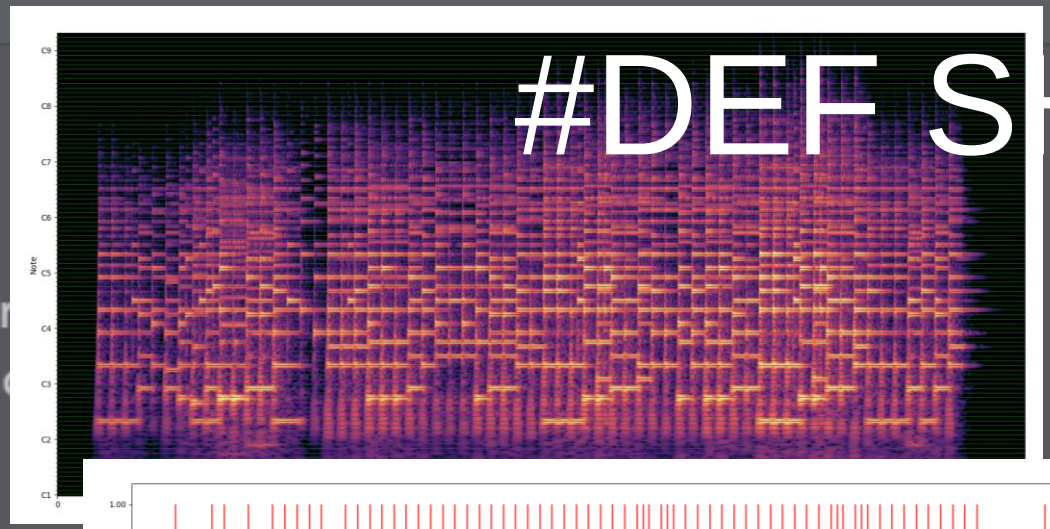
```
def
    global x; global sr; global spec_image; global cur_img
    cur_img="spec_chromaSTFT.png"
    if sr == 0:
        return
    plt.rcParams['figure.figsize'] = (22, 10.5)
    bins_per_octave = 36
    #something here
```

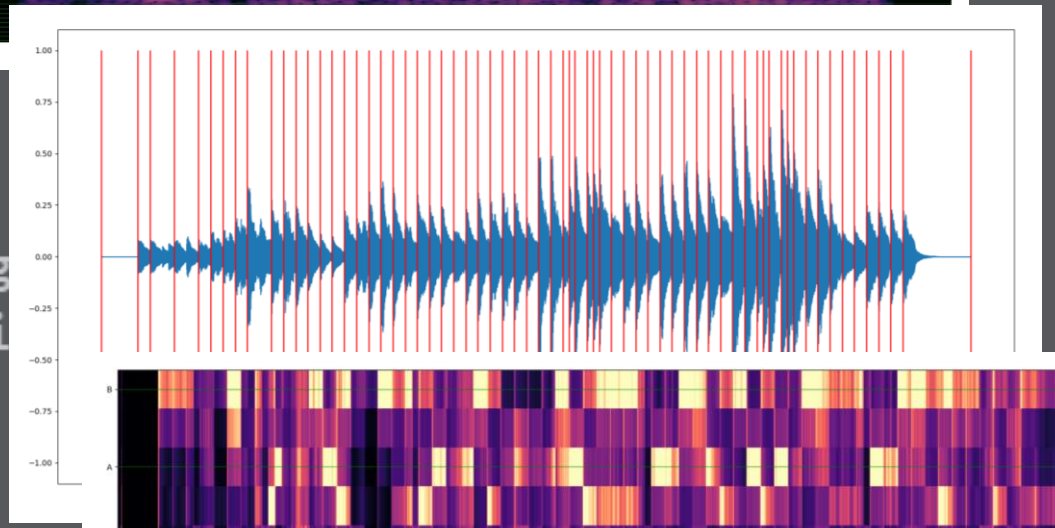
format="wav")

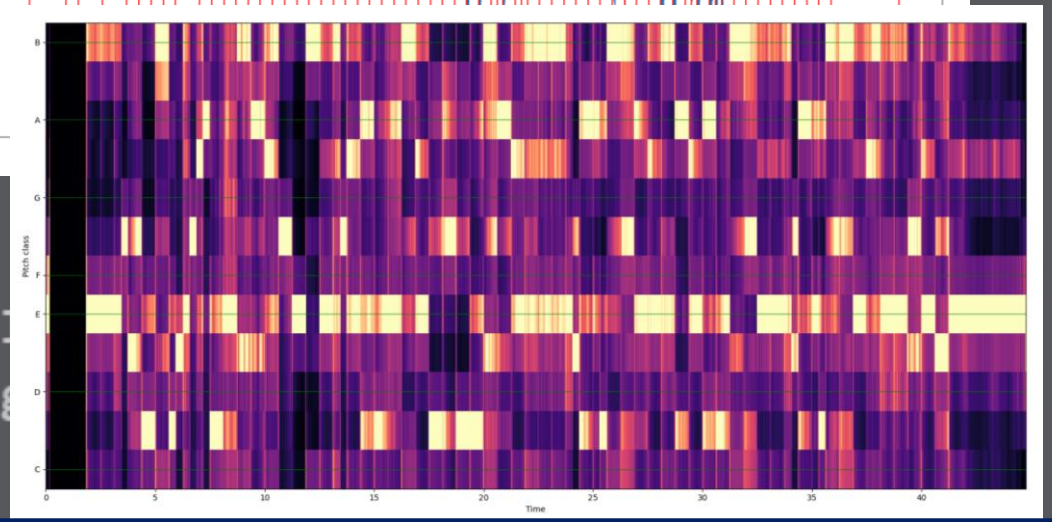
Generate the graphs with binary mask and tempo filters

驚濤心電圖 今天 13:56

```
def show_cqt():
    global cur_img
    cur_img = "spec_cqt.png"
    spec_image = ImageTk.PhotoI
    panel.configure(image = spe
    panel.image = spec_image
def show_onset():
    global cur_img
    cur_img = "onset.png"
    spec_image = ImageTk.PhotoImag
    panel.configure(image = spec_i
    panel.image = spec_image
def show_stft():
    global cur_img
    cur_img = "spec_chromaSTFT.png"
    spec_image = ImageTk.PhotoImage(
    panel.configure(image = spec_ima
    panel.image = spec_image
```



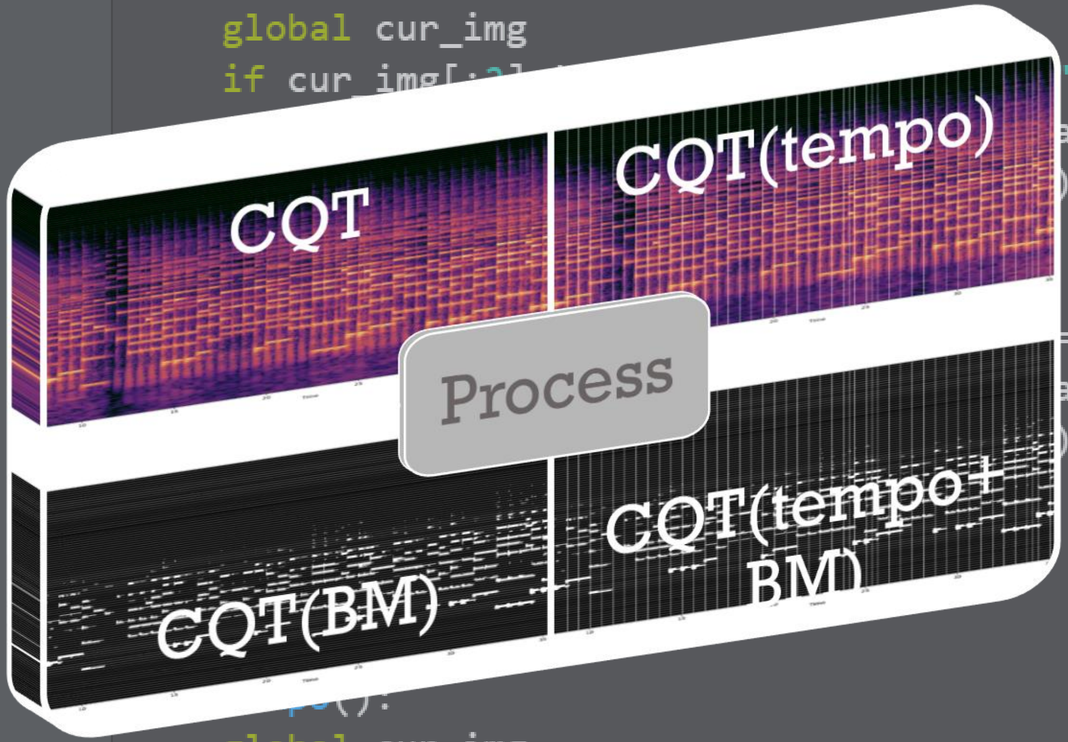




#DEF SHOW

Shows the corresponding graph of the button pressed

#DEF INFORMATION PROCESS



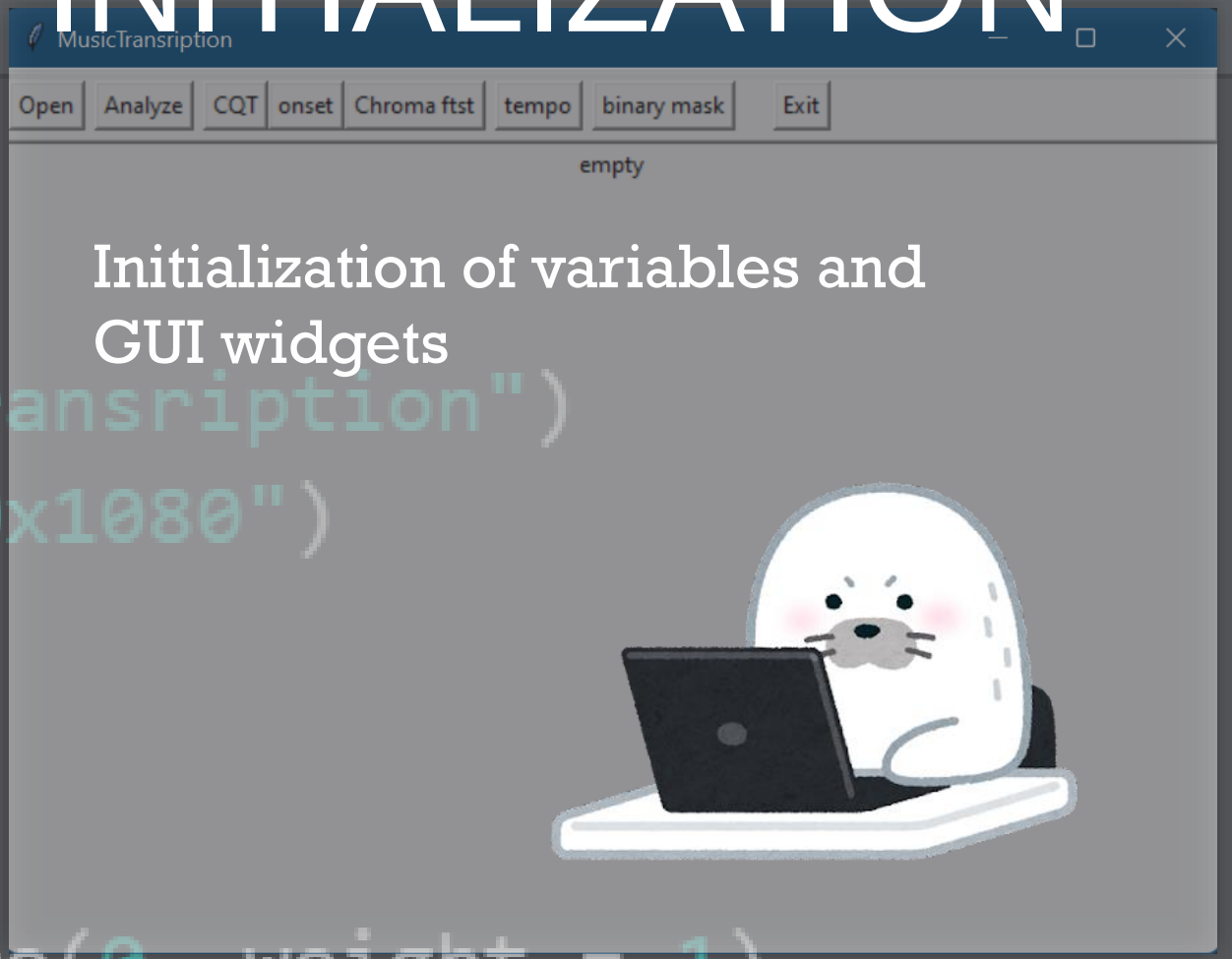

```
global cur_img
if cur_img[-9:-4] != "tempo" and cur_img != "onset.png":
    spec_image = ImageTk.PhotoImage(Image.open(cur_img[:-4]+"_tempo.png"))
    panel.configure(image = spec_image)
    panel.image = spec_image
    cur_img = cur_img[:-4]+"_tempo.png"
elif cur_img[-9:-4] == "tempo" and cur_img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

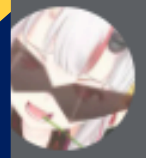
驚濤心電圖 今天 13:56

#GUI INITIALIZATION

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



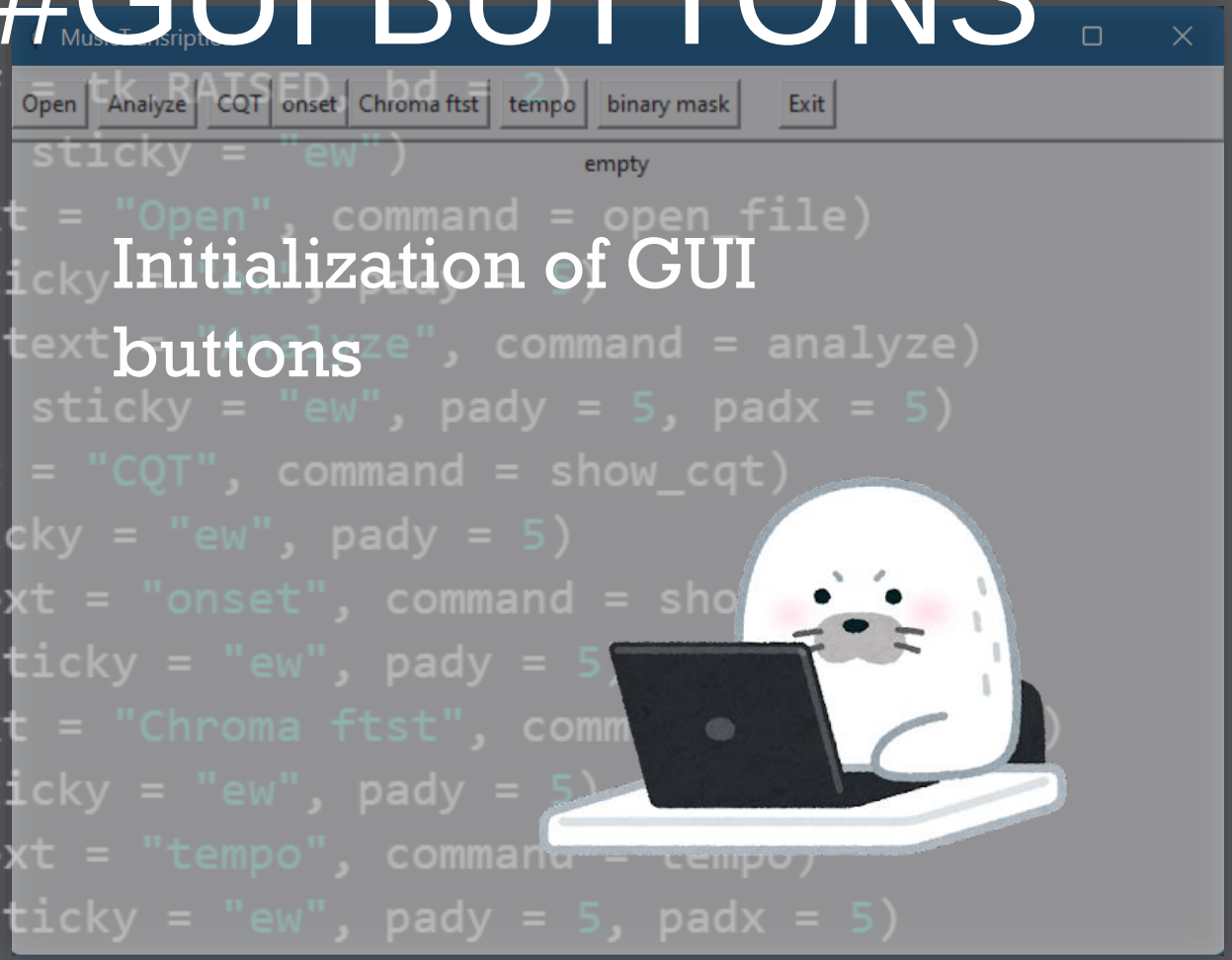
Initialization of variables and GUI widgets



驚濤心電圖 今天 13:56

#GUI BUTTONS

```
#按鈕
frm_buttons = tk.Frame(window, relief = tk.RAISED, 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)
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 = sho
btn_onset.grid(row = 0, column = 3, sticky = "ew", pady = 5,
btn_stft = tk.Button(frm_buttons, text = "Chroma ftst", comm
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)
```



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