채보 프로그램 구현

음성및오디오공학 팀프로젝트 6조

201710877 이동원

201710854 김병찬

* 개요

음성 녹음 데이터를 입력 받아 note를 추출하고, 오선보에 악보를 그리는 과정을 프로그램으로 구현.

* Note 작성 예상 결과물에 대한 유사사례 예시

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 모니터, 실내, 디스플레이이(가) 표시된 사진

자동 생성된 설명

* 기준 설정

BPM 자동 측정

피아노 음계 구간을 기준으로 (A0 ~ C8)

테스트 샘플

* + 오선보 높은음자리표에서 표현되는 구간에서 연주
  + 4/4박자
  + 피아노 연주 녹음 자료, sampling rate = 48000Hz
* 코드 구성
* import numpy as np
* import scipy.signal
* import scipy.stats
* import librosa
* import matplotlib.pyplot as plt
* import librosa.display
* import cv2

활용된 모듈 : Numpy, Scipy, Librosa, matplotlib.pyplot, OpenCV

1. 음악 파일 입력
2. ## 메인 테스트 대상은 sample0006.wav / 동요 비행기를 120BRM 메트로놈에 맞춰 Ableton Live VST, MIDI Keyboard로 연주한 녹음 파일.
3. ## 활용 VST  E\_Piano Basic.adg(Ableton Live Lite 기본 포함)
4. ## 입력 시스템 활용 가능성.
5. filepath = "C:/code/audio\_pr/sound/"
6. filename = "B120.wav"
7. ## data : wav file 읽어온 data값. 2-5.Spectogram을 통해 시간, 주파수, 음압에 대한 정보가 있음을 알 수 있다.
8. ## samplingrate : librosa.load(file, sr)에서 sr값은 파일의 샘플링 주파수다. 이 값을 받아 samplingrate 변수에 저장.
9. data , samplingrate = librosa.load(filepath + filename, sr=48000)

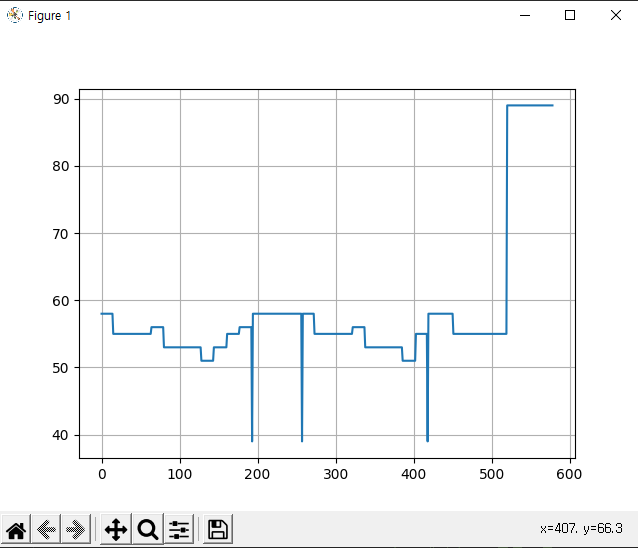
Librosa.load 활용, 음성 데이터를 활용 가능한 형태로 불러온다. 이 문서에서 활용한 데이터(B120.wav)는 BPM120으로 연주한 동요 나비야 wav file이다.

1. BPM 검출
2. ex\_tempo , \_ = librosa.beat.beat\_track(data,sr=samplingrate)
3. ## 읽기 쉽도록 반올림
4. BPM = round(ex\_tempo)
5. ## Unit\_Beat는
6. if not BPM == 0 :
7. Unit\_Beat = int((60/BPM)\*samplingrate/4)

Beat\_track 함수를 이용해 템포를 역산하고 이를 정수화해서 BPM 변수에 저장.

16분음표의 시간만큼의 sample 수를 Unit\_Beat로 저장해서 단위 형태로 뒤에서 활용.

1. 기본 주파수 검출
2. ## Unit\_Beat/8의 frame\_length로 추출했기 때문에 f0값 파생에서는 Unit\_Beat만큼의 구간내의 데이터가 8개가 된다.
3. f0 = librosa.yin(data, fmin=librosa.note\_to\_hz("A0"), fmax=librosa.note\_to\_hz("C#8"), sr=samplingrate, frame\_length=int(Unit\_Beat/16), win\_length=None, hop\_length=None)
4. ## librosa.times\_like() Return an array of time values to match the time axis from a feature matrix.
5. ## f0에 대응하는 시간 단위
6. times = librosa.times\_like(f0, sr=samplingrate)
7. ## f0에 대한 log연산으로 구간간격 1인 형태로 설정 후, numpy의 내림연산 함수 floor를 활용해 양자화
8. ## 정수값 형태로 출력 가능. A0의 음계 번호를 0으로 설정해서 피아노 연주 가능한 최소음인 A0이상의 모든음이 0 또는 양수 가 되도록 했다.
9. Q\_f0 = np.floor(12\*np.log2(f0/440)+0.5) + 48
10. ## 최빈값 검출 알고리즘인 scipy.stats.mode()를 통해 Unit\_Beat 단위로 최빈값 검출 후 note\_pitch에 저장.
11. t = 0
12. moded\_pitch = np.array([])
13. for t in range(int(len(Q\_f0)/16)) :
14. t = t + 1
15. if t < int(len(Q\_f0)/16) :
16. mode = (scipy.stats.mode(Q\_f0[16\*t:16\*(t+1)])[0])
17. moded\_pitch = np.hstack([moded\_pitch, mode])
18. plt.plot(times, moded\_pitch)
19. plt.grid(True)
20. plt.show()

moded\_pitch 의 그래프

05. note 정보 생성.

1. ## Q\_f0에서 peak 검출. Peak가 검출된 위치는 음이 연속적으로 연결되지 않는 구간이라고 볼 수 있다. 따라서 이를 기준으로 note의 길이(음표 종류)를 정할 수 있다.
2. note\_cut = np.array([])
3. peaks, properties = scipy.signal.find\_peaks(Q\_f0, height=0)
4. print(peaks)
5. #u = 0
6. #for u in range(len(peaks)) :
   * 1. #u = u + 1
7. note\_cut = np.around(peaks/16) ## 내림연산
8. note\_cut = note\_cut.astype(int) ## int변환
9. print(note\_cut)

[ 11 17 258 260 267 518 520 524 529 1032 1034 1036 1041 1045

1050 1291 1299 1426 1553 1556 1560 1564 1816 2067 2070 2073 2077 2082

2325 2327 2332 2584 2591 2596 2601 2608 2615 2622 2629 2840 2846 2973

3099 3108 3113 3119 3356 3362 3366 3491 3615 3619 3624 4129 4131 4134

4141 4387 4391 4395 4400 4406 4647 4649 4653 4658 4905 4911 4914 4917

5163 5165 5170 5174 5177 5180 5302 5421 5424 5427 5432 5435 5438 5678

5682 5685 5689 5693 6194 6196 6200 6202 6206 6211 6217 6454 6459 6462

6466 6470 6473 6481 6711 6721 6726 6969 6979 6984 7107 7228 7230 7235

7485 7488 7493 7744 7751 8324]

* print(peaks)의 실행 결과

[ 0 1 16 16 16 32 32 32 33 64 64 64 65 65 65 80 81 89

97 97 97 97 113 129 129 129 129 130 145 145 145 161 161 162 162 163

163 163 164 177 177 185 193 194 194 194 209 210 210 218 225 226 226 258

258 258 258 274 274 274 275 275 290 290 290 291 306 306 307 307 322 322

323 323 323 323 331 338 339 339 339 339 339 354 355 355 355 355 387 387

387 387 387 388 388 403 403 403 404 404 404 405 419 420 420 435 436 436

444 451 451 452 467 468 468 484 484 520]

* print(note\_cut)의 실행 결과

06. 구축된 data를 통해 악보 작성 시각화하는 알고리즘

imgpath = "C:/code/audio\_pr/score/"

imgfile = "emptyscore.png"

## 공백 오선보 샘플 load

score = cv2.imread(imgpath + imgfile, cv2.IMREAD\_GRAYSCALE)

## note 종류별로 image load해서 좌표계에 띄울 수 있도록 준비

def setwholenote(hpos, vpos) :

        Xsize = 30

        Ysize = 22

        whole = cv2.imread(imgpath + "wholenote.png", cv2.IMREAD\_GRAYSCALE)

        note\_whole = cv2.resize(whole, dsize=(Xsize,Ysize), interpolation=cv2.INTER\_AREA)

        #rows, cols, channels = wholenote.shape

        ROI = score[vpos : Ysize + vpos, hpos : Xsize + hpos]

        DST = cv2.bitwise\_or(ROI, note\_whole)

        score[vpos : Ysize + vpos, hpos : Xsize + hpos] = DST

def sethalfnote(hpos, vpos) :

        Xsize = 30

        Ysize = 94

        half = cv2.imread(imgpath + "halfnote.png", cv2.IMREAD\_GRAYSCALE)

        note\_half = cv2.resize(half, dsize=(Xsize,Ysize), interpolation=cv2.INTER\_AREA)

        ROI = score[vpos : Ysize + vpos, hpos : Xsize + hpos]

        DST = cv2.bitwise\_or(ROI, note\_half)

        score[vpos : Ysize + vpos, hpos : Xsize + hpos] = DST

반복해서 note생성 과정을 종류별로 구현한다.

거의 같은 형태이므로 분량상 생략.

sample\_note\_start = np.array([0,16,32, 64,80,96, 128,144,160,176,192,208,224, 256,272,288,304,320,336,352, 384,400,416,432,448,464,496])

sample\_note\_pitch = np.array([46,43,43, 44,41,41, 39,41,43,44,46,46,46, 46,43,43,43,44,41,41, 39,43,46,46,43,43,43])

sample\_note\_time = np.array([16,16,32, 16,16,32, 16,16,16,16,16,16,32, 16,16,16,16,16,16,32, 16,16,16,16,16,16,32])

start = sample\_note\_start

pitch = sample\_note\_pitch

time = sample\_note\_time

start는 note의 시작 지점, pitch는 음값, time은 지속 시간이 된다.

현 시점 기준 구현 내용 끝.