WIRELESS SOUND CONTROL:

❖ A Sound Project Using Python:

➢ Abstract:

This Paper provides the implementation of **Various Digital Audio Effects (DAFXS)** as a combination of user defined parameters and input sound signal. An approach to implement various effects like delay-based effects, spatial effects, time varying and modulator is provided. A unique listening environment is provided using 3-D spatialization and localization, simulated surround sound, dialogue normalization, dynamic range control and down-mixing. An attempt has also been made for music and voice separation provide karaoke effect to the sound.

All the implementations are provided in python which is widely adopted, open source and general-purpose programming language and has a vast array of code libraries and development tools, and integrates well with many other programming languages, frameworks and musical applications.

➢ INTRODUCTION:

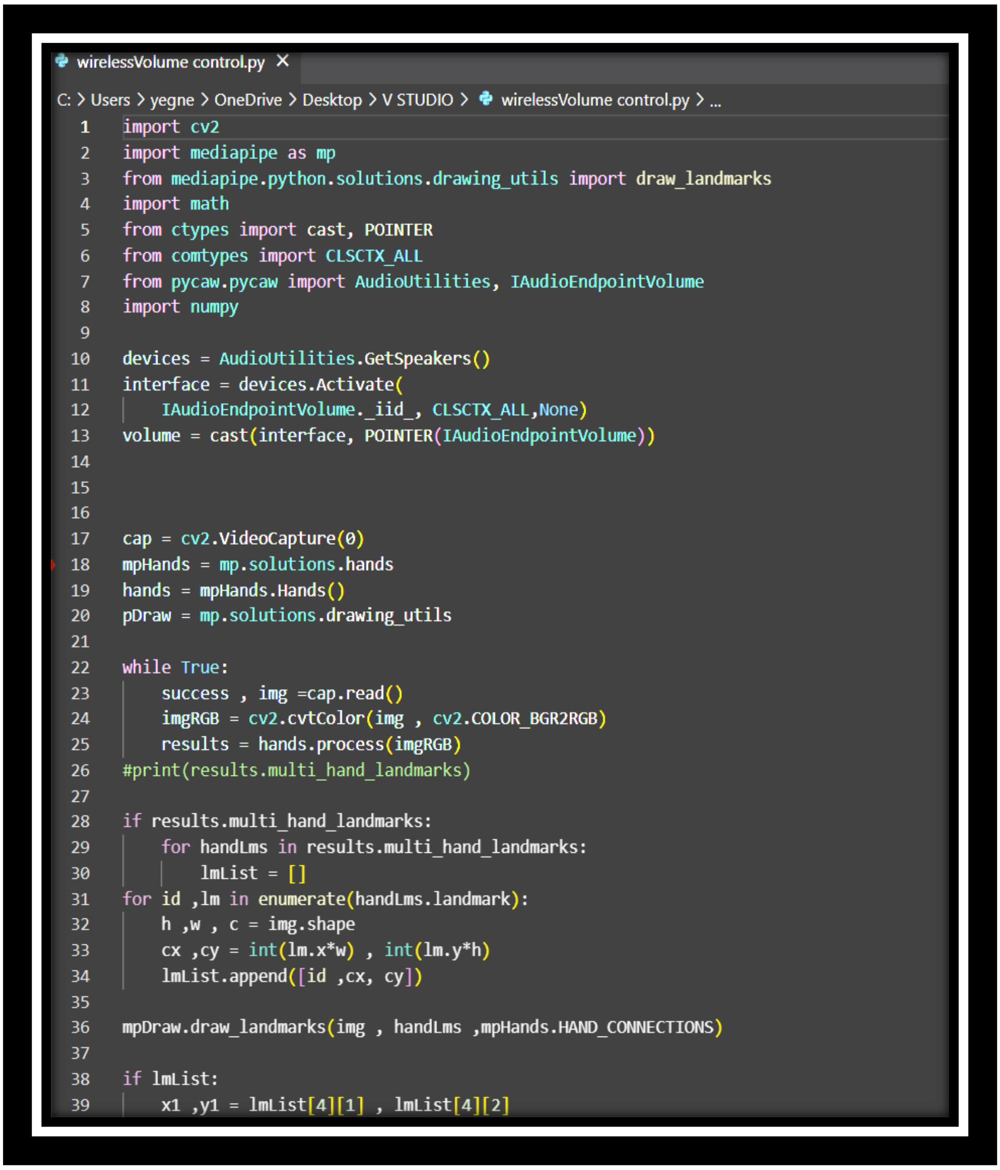
This doc has been written in support of a sound project which largely encircles the domain of digital audio effects -DAFX as an acronym. The Python Programming language has been chosen for development of this project due to its apprehensive provision of sound API's, tools, libraries and support for audio signal processing and its comprehensive ability to integrate with other programming constructs. At an abstract level this project comprises of two modules audio effects and dialogue normalization coupled with dynamic range control. The concept underlying dialog normalization and dynamic range control goes hand in hand. Consider for e. g. a case of TV channels where each channel sounds different although all of them have been set at the same volume. Likewise, a portion of an audio signal may also sound different from another one although both portions ae played at the same volume. Hence, the challenge here lies in controlling and modifying values of chunks in each portion sound similar when played at the same volume. A code for same has been implemented in this project.

Digital audio effects (DAFX) are systems that modify audio signals. These transformations are made according to some control parameters the premise and delivers output sounds. The audio effects are being an extensively vast module in itself can the inherent property of audio signals that is modified by each effects as:

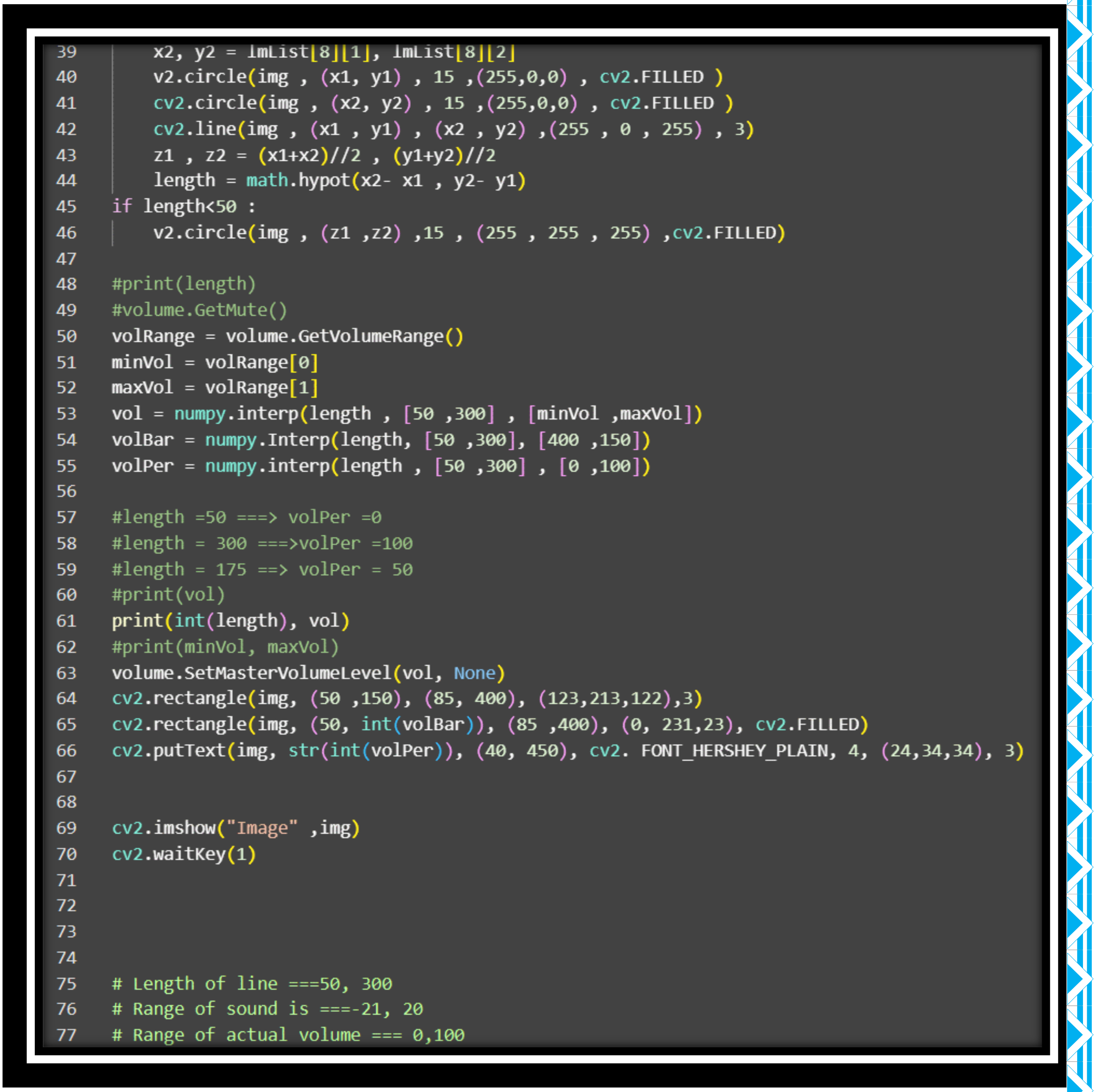
- Delay based effects – Chorus, flanger, vibrato etc.
- Spatial effects – reverberation, panning, 3D effect for headphones using hrtf.
- Time varying filters – wah-wah.
- Modulators – tremolo.
- Karaoke using audio downmixing.
- Other exciting effects.

In order to reduce the restrictions on the type of audio file that can be scanned as input, a provision for audio transcoding and compression of a .wav file has also been provided for a dedicated attempt has been made to create and integrate a comprehensive listening and sound manipulating environment in a single project as standalone software which provides for uniqueness to the project.

➢ CODE:

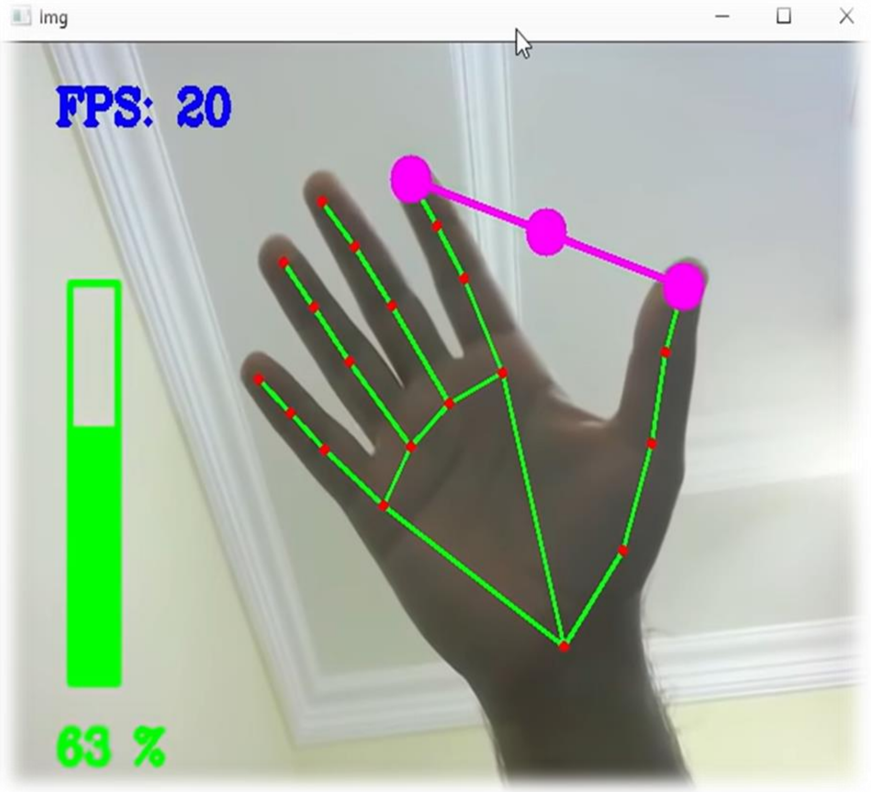

wirelessVolume control.py ×
C: > Users > yegne > OneDrive > Desktop > V STUDIO > wirelessVolume control.py > ...

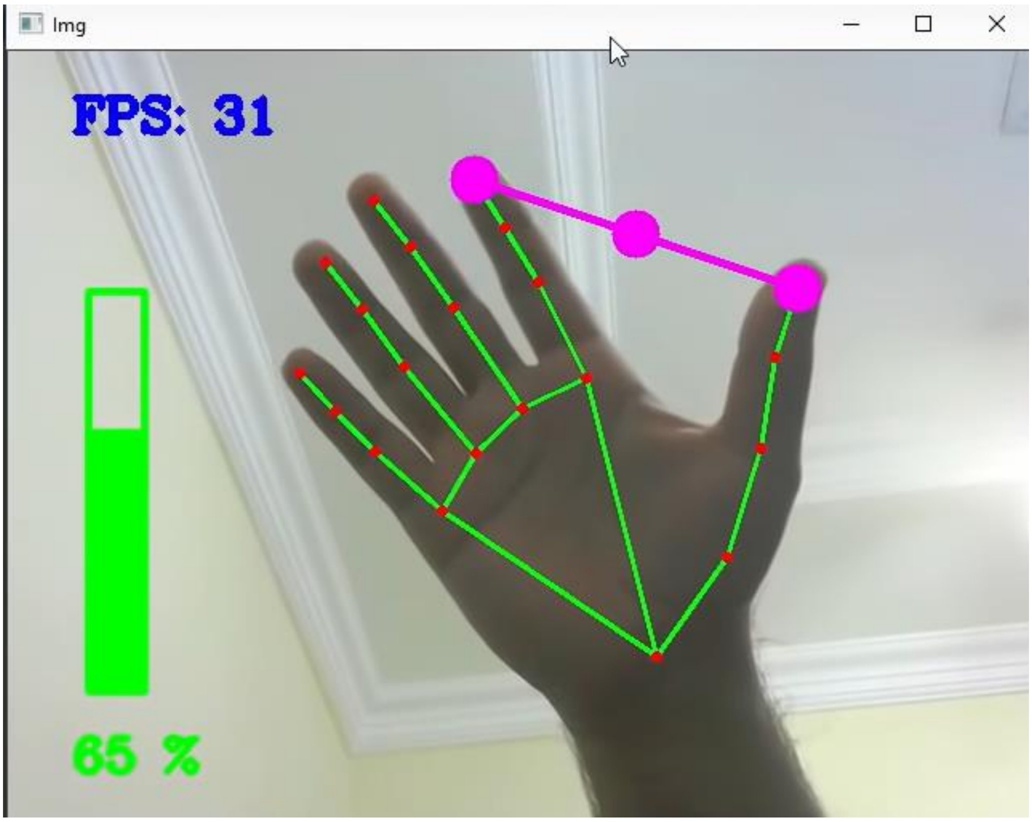
```python
import cv2
import mediapipe as mp
from mediapipe.python.solutions.drawing_utils import draw_landmarks
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy

devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL,None)
volume = cast(interface, POINTER(IAudioEndpointVolume))



cap = cv2.VideoCapture(0)
mpHands = mp.solutions.hands
hands = mpHands.Hands()
pDraw = mp.solutions.drawing_utils

while True:
    success , img =cap.read()
    imgRGB = cv2.cvtColor(img , cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
#print(results.multi_hand_landmarks)

if results.multi_hand_landmarks:
    for handLms in results.multi_hand_landmarks:
        lmList = []
for id ,lm in enumerate(handLms.landmark):
    h ,w , c = img.shape
    cx ,cy = int(lm.x*w) , int(lm.y*h)
    lmList.append([id ,cx, cy])

mpDraw.draw_landmarks(img , handLms ,mpHands.HAND_CONNECTIONS)

if lmList:
    x1 ,y1 = lmList[4][1] , lmList[4][2]
```



```python
    x2, y2 = lmList[8][1], lmList[8][2]
    v2.circle(img , (x1, y1) , 15 ,(255,0,0) , cv2.FILLED )
    cv2.circle(img , (x2, y2) , 15 ,(255,0,0) , cv2.FILLED )
    cv2.line(img , (x1 , y1) , (x2 , y2) ,(255 , 0 , 255) , 3)
    z1 , z2 = (x1+x2)//2 , (y1+y2)//2
    length = math.hypot(x2- x1 , y2- y1)
if length<50 :
    v2.circle(img , (z1 ,z2) ,15 , (255 , 255 , 255) ,cv2.FILLED)

#print(length)
#volume.GetMute()
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = numpy.interp(length , [50 ,300] , [minVol ,maxVol])
volBar = numpy.Interp(length, [50 ,300], [400 ,150])
volPer = numpy.interp(length , [50 ,300] , [0 ,100])

#length =50 ===> volPer =0
#length = 300 ===>volPer =100
#length = 175 ==> volPer = 50
#print(vol)
print(int(length), vol)
#print(minVol, maxVol)
volume.SetMasterVolumeLevel(vol, None)
cv2.rectangle(img, (50 ,150), (85, 400), (123,213,122),3)
cv2.rectangle(img, (50, int(volBar)), (85 ,400), (0, 231,23), cv2.FILLED)
cv2.putText(img, str(int(volPer)), (40, 450), cv2. FONT_HERSHEY_PLAIN, 4, (24,34,34), 3)


cv2.imshow("Image" ,img)
cv2.waitKey(1)




# Length of line ===50, 300
# Range of sound is ===-21, 20
# Range of actual volume === 0,100
```

➢OUTPUT:





➢ CONCLUSION:

The audio effects implemented in this project have been observed to have considerable accuracy. The karaoke effect has a stencil of vocals left in the final output which ideally should be completely eliminated. Thus, complete elimination of vocals herein should be considered as future scope of the project. Also, optimization in the implementation of dynamic range control should be considered in the future scope. All effects promised in this document have been implemented and delivered, the accuracy of each varying to a certain extent. My references are **DAFX- Digital Audio Effects.**