pyrecspecwaterfall

March 12, 2017

1 Pyrecspecwaterfall

Using Pyaudio, record sound from the audio device and plot a waterfall spectrum display, for 8 seconds.

Usage example: python pyrecspecwaterfall.py

```
- Gerald Schuller, November 2014
```

- **Import the relevant modules:**

```
In [1]: import pyaudio
        import struct
        import numpy as np
        import cv2
```

- **Define the variables:**

```
In [2]: CHUNK = 2048 #Blocksize
        WIDTH = 2 #2 bytes per sample
        CHANNELS = 1 #2
        RATE = 48000  #Sampling Rate in Hz
```

- **Initialise the sound card print out its detail:**

```
In [3]: p = pyaudio.PyAudio()

        a = p.get_device_count()
        print("device count=",a)

        for i in range(0, a):
            print("i = ",i)
            b = p.get_device_info_by_index(i)['maxInputChannels']
            print(b)
            b = p.get_device_info_by_index(i)['defaultSampleRate']
            print(b)

        stream = p.open(format=p.get_format_from_width(WIDTH),
                        channels=CHANNELS,
                        rate=RATE,
                        input=True,
                        output=True,
                        #input_device_index=3,
                        frames_per_buffer=CHUNK)


        print("* recording")
```

```
('device count=', 11L)
('i = ', 0)
2
44100.0
('i = ', 1)
0
44100.0
('i = ', 2)
128
48000.0
('i = ', 3)
0
44100.0
('i = ', 4)
0
44100.0
('i = ', 5)
0
44100.0
('i = ', 6)
0
44100.0
('i = ', 7)
32
44100.0
('i = ', 8)
0
48000.0
('i = ', 9)
32
44100.0
('i = ', 10)
16
44100.0
* recording
```

- **Size of waterfall diagram:**
- **max CHUNK/2 rows:**

```
In [4]: rows = 500
        cols = 512
        fftlen = cols * 2
        frame = 0.0 * np.ones((rows, cols, 3))
```

- **Start recording and plotting the real time spectrogram waterfall(upward) which shows the intensities but the colours in the row and time vertically moving upward. The steps involve the block wise prcessing of the recorded samples and then show its spectrogram in row.**

```
In [5]: ctr = 0
        while(True):
            ctr = ctr + 1
            #Reading from audio input stream into data with block length "CHUNK":
            data = stream.read(CHUNK)
            #Convert from stream of bytes to a list of short integers (2 bytes here) in "samples
            #shorts = (struct.unpack( "128h", data ))
            shorts = (struct.unpack( 'h' * CHUNK, data ));
            samples = np.array(list(shorts), dtype=float);

            if (ctr%4 == 0):
                #shift "frame" 1 up:
                frame[0:(rows-1),:]=frame[1:rows,:];
                #compute magnitude of 1D FFT of sound
                #with suitable normalization for the display:
                #frame=np.abs(np.ffqt.fft2(frame[:,:,1]/255.0))/512.0
                #write magnitude spectrum in lowes row of "frame":
                R = 0.25 * np.log((np.abs(np.fft.fft(samples[0:fftlen])[0:(fftlen/2)]/np.sqrt(ff
                #Color mapping:
                #Red:
                frame[rows-1,:,2] = R
                #Green:
                frame[rows-1,:,1] = np.abs(1-2*R)
                #Blue:
                frame[rows-1,:,0] = 1.0-R
                #frame[rows-1,:,0]=frame[rows-1,:,1]**3
                # Display the resulting frame
                cv2.imshow('frame',frame)
            #Keep window open until key 'q' is pressed:
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
```

- **When everything's done then release the capture:**

```
In [6]: cv2.destroyAllWindows()

        stream.stop_stream()
        stream.close()
        p.terminate()
```