07_SeeingSound

March 7, 2016

In [14]: import matplotlib
import scipy
from __future__ import division
import scikits.audiolab as audio
import numpy
import matplotlib.pyplot as plt
import pylab
from numpy import *

%matplotlib inline

def setup_graph(title='', x_label='', y_label='', fig_size=None):
    fig = plt.figure()
    if fig_size != None:
        fig.set_size_inches(fig_size[0], fig_size[1])
    ax = fig.add_subplot(111)
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)

1 Seeing sound!

In [15]: # NOTE: This is only works with 1 channel (mono). To record a mono audio sample,
    # you can use this command: rec -r 44100 -c 1 -b 16 test.wav
    (input_signal, sample_rate, bits) = audio.wavread("audio_files/vowel_ah.wav")
    time_array = arange(0, len(input_signal)/sample_rate, 1/sample_rate)

In [16]: setup_graph(title='Ah vowel sound', x_label='time (in seconds)', y_label='amplitude', fig_size=(14,7))
   _ = plot(time_array[0:4000], input_signal[0:4000])
In [17]: fft_out = numpy.fft.rfft(input_signal)
    fft_mag = [sqrt(i.real**2 + i.imag**2)/len(fft_out) for i in fft_out]
    num_samples = len(input_signal)
    rfreqs = [(i*1.0/num_samples)*sample_rate for i in range(num_samples//2+1)]

    setup_graph(title='FFT of Ah Vowel (first 5000)', x_label='FFT Bins', y_label='magnitude', fig,
    _ = plot(rfreqs[0:5000], fft_mag[0:5000])
1.1 Few notes about this

• Ratio of harmonics = timbre = This is what makes different people’s voices sound different (and different from violins)
  – Even if they are singing the same note, and the same volume
• Possible application: synthesizing new sounds (with different harmonic profiles)
  – Example: If you changed the ratio of the harmonics, you could make your voice sound like something else (Darth Vader?)

2 Spectrogram (FFT over time)

2.0.1 Axes

• x-axis: time
• y-axis: frequency
• z-axis (color): strength of each frequency

2.0.2 See the Harmonics!

In [18]: (doremi, doremi_sample_rate, bits) = audio.wavread("audio_files/do-re-mi.wav")

In [19]: setup_graph(title='Spectrogram of diatonic scale (44100Hz sample rate)', x_label='time (in seconds)', y_label='frequency', fig_size=(14,8))

_ = pylab.specgram(doremi, Fs=doremi_sample_rate)

In [20]: doremi_8000hz = [doremi[i] for i in range(0, len(doremi), 44100//8000)]

setup_graph(title='Spectrogram (8000Hz sample rate)', x_label='time (in seconds)', y_label='frequency', fig_size=(14,7))

_ = pylab.specgram(doremi_8000hz, Fs=8000)
In [21]: doremi_4000hz = [doremi[i] for i in range(0, len(doremi), 44100//4000)]
setup_graph(title='Spectrogram (4000Hz sample rate)', x_label='time (in seconds)', y_label='frequency', fig_size=(14,7))
_ = specgram(doremi_4000hz, Fs=4000)

---------------------------------------------------------------------------
NameError Traceback (most recent call last)
<ipython-input-21-c673d80a02d9> in <module>()
1 doremi_4000hz = [doremi[i] for i in range(0, len(doremi), 44100//4000)]
2 setup_graph(title='Spectrogram (4000Hz sample rate)', x_label='time (in seconds)', y_label='frequency', fig_size=(14,7))
----> 3 _ = specgram(doremi_4000hz, Fs=4000)

NameError: name 'specgram' is not defined
2.1 A few things to note

- Something that sounds like a single note actually is made up of a bunch of harmonics
- Harmonics are integer multiples of the fundamental frequency
- Notice that the spacing between the harmonics of the first note is about double of the spacing between the harmonics in the last note (1 octave difference)

In [ ]: