디지털 신호 처리 7주차 과제

201710758 휴먼지능정보공학과 김진성

In [13]:
```
import os

if not os.path.exists('../thinkdsp.py'):
    !wget https://github.com/AllenDowney/ThinkDSP/raw/master/code/thinkdsp.

import numpy as np
import matplotlib.pyplot as plt
from thinkdsp import read_wave
from thinkdsp import decorate
```

챕터5의의 autocorr을 사용해, lag별 corr을 뽑아낸 다음, 약간의 센스(argmax)를 사용해 corr을 최대로 갖는 lag을 뽑아낸다음, framerate에 나누어 preiod를 구해서, 최종적으로 frequency를 구해내는 함수를 estimate_fundamental이라는 함수로 만들어 보겠습니다.

우선 autocorr를 해주는 함수를 만들어 줍니다.

In [14]:
```
def serial_corr(wave, lag=1):
    n = len(wave)
    y1 = wave.ys[lag:]
    y2 = wave.ys[:n-lag]
    corr_mat = np.corrcoef(y1, y2)
    return corr_mat[0, 1]
```

In [15]:
```
def autocorr(wave):
    lags = np.arange(len(wave.ys)//2)
    corrs = [serial_corr(wave, lag) for lag in lags]
    return lags, corrs
```

위에서 설명한대로, estimate_fundamental 함수를 만들어 줍니다.

In [16]:
```
def estimate_fundamental(segment, low, high):
    lags, corrs = autocorr(segment)
    lag = np.array(corrs[low:high]).argmax() + low
    period = lag / segment.framerate
    frequency = 1 / period
    return frequency
```

잘 작동하는지 보는 테스트 단계 입니다.

chap5에서 제공해주는 고주파에서 저주파로 가는 소리를 가져와 봤습니다.

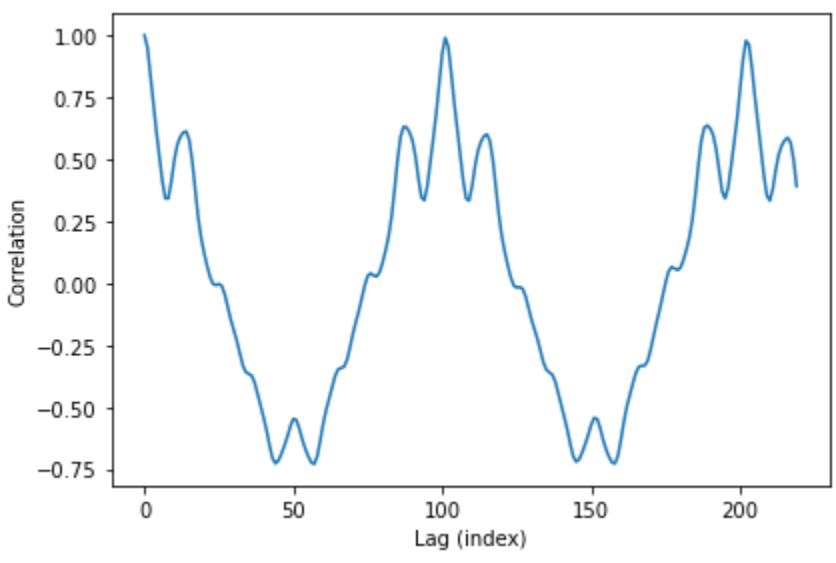
In [18]:
```
from thinkdsp import read_wave

wave = read_wave('../28042__bcjordan__voicedownbew.wav')
wave.normalize()
wave.make_audio()
```

Out[18]:

In [21]:
```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
# freq = estimate_fundamental(segment)
# freq

lags, corrs = autocorr(segment)
plt.plot(lags, corrs)
decorate(xlabel='Lag (index)', ylabel='Correlation')
```



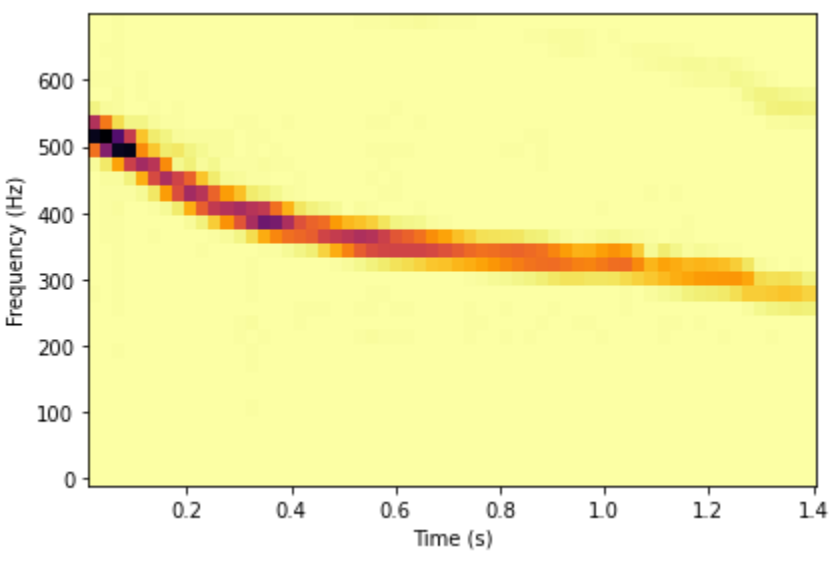
In [33]:
```
duration = 0.01
segment = wave.segment(start=0.2, duration=duration)
freq = estimate_fundamental(segment,low=70,high=150)
freq
```

Out[33]:
```
436.63366336633663
```

알아서 최대 corr을 갖는 lag이 계산되어서 period -> freq순으로 도출되었습니다!

아래 그래프에다가 제가 만든 estimate_fundamental 함수를 사용해 0.0~1.4초를 0.05 step마다 frquency를 구해서 그려서, 얼마나 정확하게 도출해내는지 봐보겠습니다!

In [24]:
```
wave.make_spectrogram(2048).plot(high=700)
decorate(xlabel='Time (s)',
         ylabel='Frequency (Hz)')
```

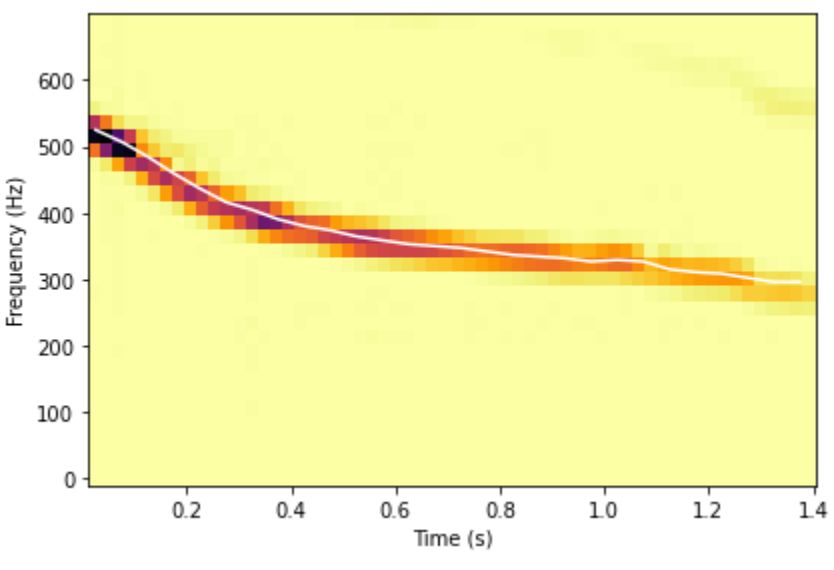


In [28]:
```
step = 0.05
starts = np.arange(0.0, 1.4, step)

ts = []
freqs = []

for start in starts:
    ts.append(start + step/2)
    segment = wave.segment(start=start, duration=duration)
    freq = estimate_fundamental(segment,low=70,high=150)
    freqs.append(freq)
```

In [32]:
```
wave.make_spectrogram(2048).plot(high=700)
plt.plot(ts, freqs, color='white')
decorate(xlabel='Time (s)',
                    ylabel='Frequency (Hz)')
```



매우 근사하게 잡아내는걸 확인할 수 있었습니다!