Lecture 6: Music

Mark Hasegawa-Johnson

ECE 401: Signal and Image Analysis, Fall 2020

Outline

Two-sided spectrum

The **spectrum** of $x(t)$ is the set of frequencies, and their associated phasors,

$$\text{Spectrum}\left(x(t)\right) = \{(f_{-N}, a_{-N}), \ldots, (f_0, a_0), \ldots, (f_N, a_N)\}$$

such that

$$x(t) = \sum_{k=-N}^{N} a_k e^{j2\pi f_k t}$$

Summary

- **Fourier Analysis** (finding the spectrum, given the waveform):

$$X_k = \frac{1}{T_0}\int_0^{T_0} x(t) e^{-j2\pi kt/T_0} dt$$

- **Fourier Synthesis** (finding the waveform, given the spectrum):

$$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{j2\pi kt/T_0}$$

- **DFT Analysis** (finding the spectrum, given the waveform):

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

- **DFT Synthesis** (finding the waveform, given the spectrum):

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}$$

Outline

Pythagorean Tuning

- Humans have always known that $f_2 = 2f_1$ (length of one string is twice the length of the other) means they are an octave apart ("same note").
- A 3:2 ratio ($f_2 = 1.5f_1$) is a musical perfect fifth.
- Pythagoras is attributed with a system of tuning that created an 8-note scale by combining 3:2 and 2:1 ratios ("Pythagorean tuning"), used in some places until 1600.

Equal-Tempered Tuning

Equal-tempered tuning divides the octave into twelve equal ratios.

- **Semitones:** the number of semitones, s, separating two tones f_2 and f_1 is given by

$$s = 12\log_2\left(\frac{f_2}{f_1}\right)$$

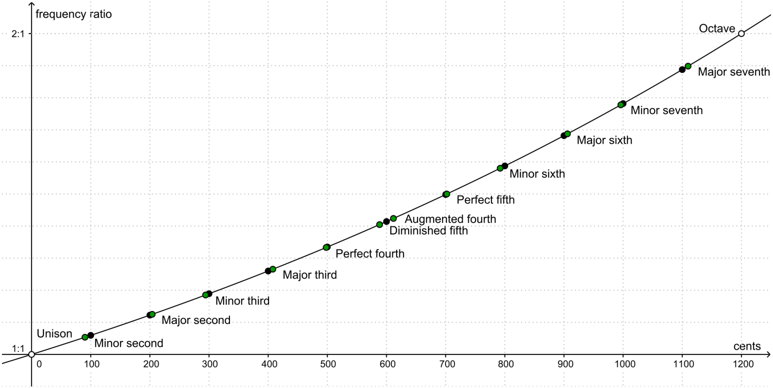
- **Cents:** the number of cents, n, separating two tones f_2 and f_1 is given by

$$n = 1200\log_2\left(\frac{f_2}{f_1}\right)$$

Pythagorean vs. Equal-Tempered Tuning

Pythagorean, Equal-Tempered, and Just Intonation

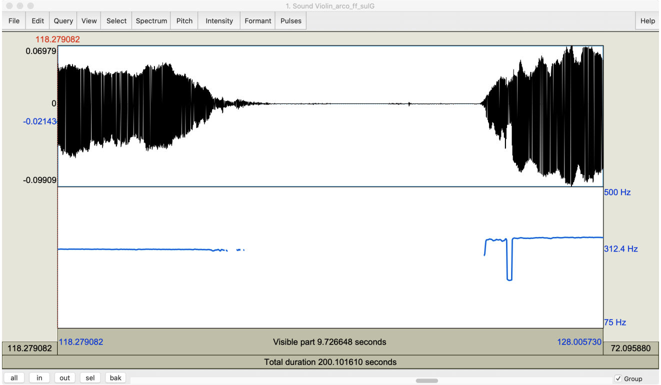
Pythagorean vs. Equal-Tempered Tuning



By SharkD, public domain image, `https://commons.wikimedia.org/wiki/File:Music_intervals_frequency_ratio_equal_tempered_pythagorean_comparison.svg`

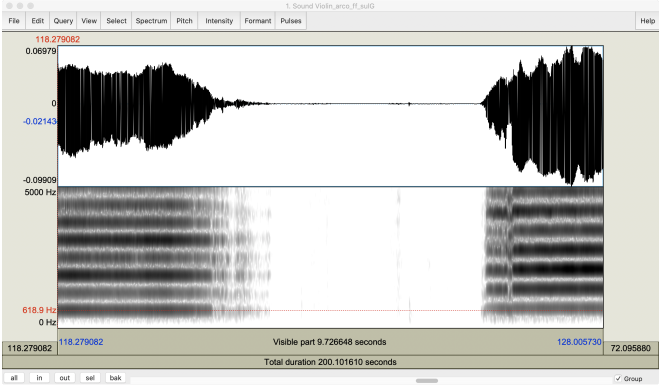
Outline

Pitch Tracking: Intended Output

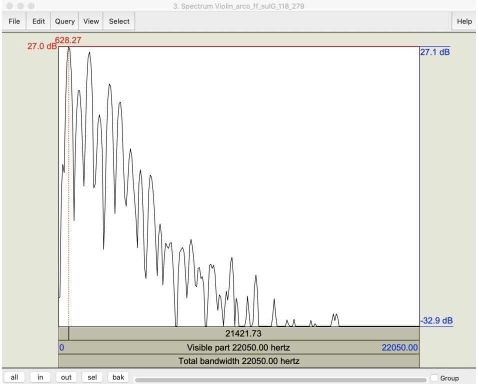


Pitch Tracking: Input

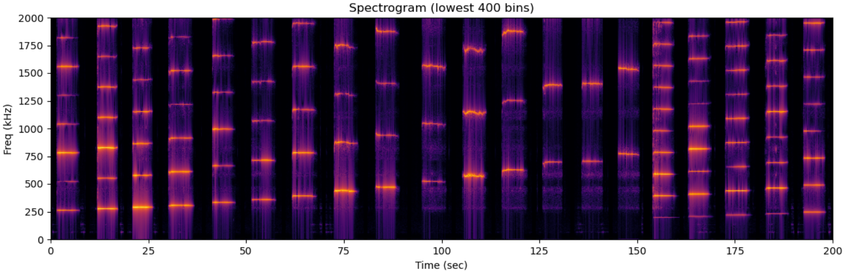
Violin.arco.ff.sulG.C4Eb5.aiff

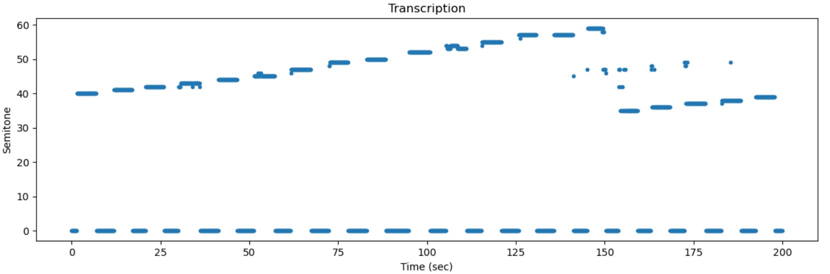


Pitch Tracking: One frame of the input looks like this

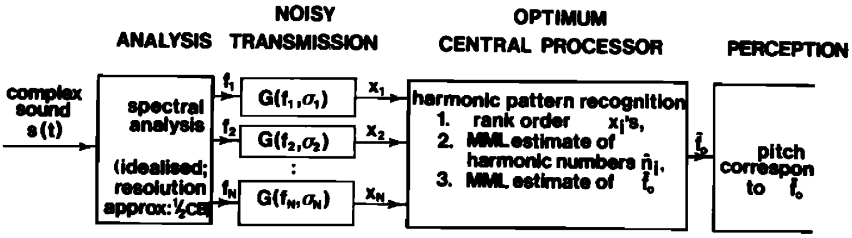


Pitch Tracking: Input and Output





The Harmonic Sieve Algorithm: Overview



(c) Duifhuis, Willems & Sluyter, J. Acoust. Soc. Am. 71(6):1568-1580, 1982

The Harmonic Sieve

1. Compute the DFT, $X[k]$.
2. For any given frequency f, define the energy at that frequency to include all of the magnitude DFT within $\pm 5\%$, i.e.,

$$E(f) = \sum_{k=0.95(Nf/F_s)}^{(1.05)Nf/F_s} |X[k]|$$

3. In order to test the Goodness of F_0 as a possible pitch frequency, add up the energy of its first 11 harmonics:

$$G(F_0) = \sum_{h=1}^{11} E(hF_0)$$

4. Choose the pitch with the best goodness.

The Harmonic Sieve

- Notice that the 11 harmonic frequencies are given by:

$$\vec{f} = [F_0, 2F_0, 3F_0, \ldots, 11F_0] = F_0 \times [1, 2, 3, \ldots, 11]$$

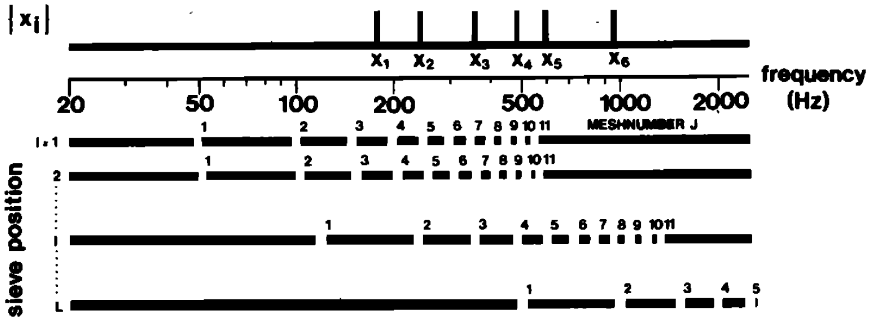
- Duifhuis, Willems & Sluyter had the clever idea of computing pitch on a semitone scale:

$$S(\vec{f}) = 12\log_2(\vec{f}) = S(F_0) + \vec{M}$$

So you can search all of the 88 keys on a piano by starting with $S = 0$ (the lowest note, A0), and searching all the way up to S=87 (the highest note, C8). For each one, you just add the harmonic sieve to get the frequencies of all the harmonics:

$$\vec{M} = [12\log_2(1), 12\log_2(2), 12\log_2(3), \ldots, 12\log_2(11)]$$

The Harmonic Sieve



(c) Duifhuis, Willems & Sluyter, J. Acoust. Soc. Am. 71(6):1568-1580, 1982

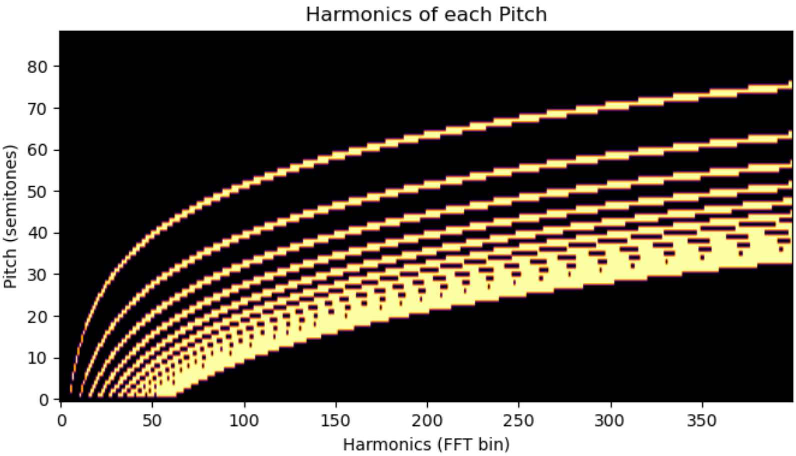
Figuring out which bins to average

So to figure out which bins to average, for any given pitch F_0:

1. Add the pitch in semitones ($S(F_0)$) to the mask ($\vec{M}$).
2. Convert back into linear frequency:

$$k = \left(\frac{N}{F_s}\right) 2^{(S(F_0)+\vec{M})/12}$$

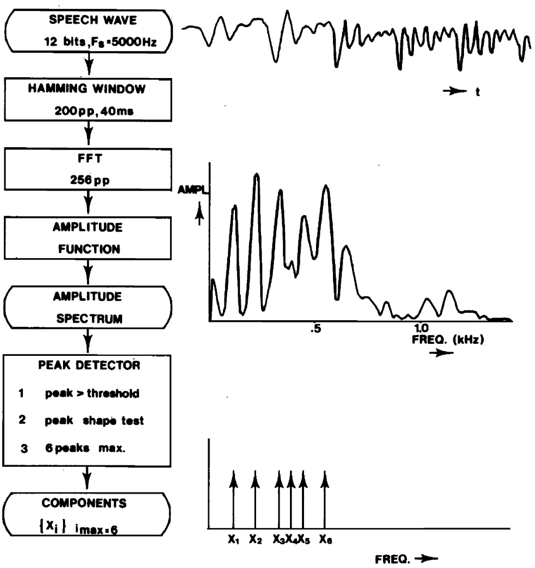
Masks for each of the notes on the piano



Duifhuis-Willems-Sluyter Spectral Analysis

- Duifhuis, Willems & Sluyter also used a "peak detector" step, after their amplitude spectrum, in order to extract peaks from the spectrum before they applied the sieve.
- I found this step to be unnecessary when I was designing MP2.
- On the other hand, this kind of peak detection is used used by Shazam, Soundhound, Beatfind, Google Sound Search etc., to reduce the number of bits per song, so that they can efficiently identify the song you're listening to. So you might find that part of the Duifhuis et al. article interesting, even though we're not using it in MP2.

Duifhuis-Willems Spectral Analysis



(c) Duifhuis, Willems & Sluyter, J. Acoust. Soc. Am. 71(6):1568-1580, 1982

Outline

Fourier Synthesis

Suppose you know $X[k]$. How can you get $x[n]$ back again? That's right!

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N}$$

Fourier Synthesis without phase

Suppose you know the **magnitude only**, $|X[k]|$. How can you get $x[n]$ back again?

- Pretend that $\angle X[k] = 0$. Oops. Sounds like a click.
- Pretend that $\angle X[k]$ is a random number between 0 and 2π. Oops. Sounds like noise.
- Be smart about the relationship between frequency and phase.

The relationship between frequency and phase

Notice that we could write

$$A \cos(2\pi f t + \theta) = A \cos(\phi(t))$$

where $\phi(t)$ is the instantaneous phase:

- at time $t = 0$, the instantaneous phase is just

$$\phi(0) = \theta$$

- At the end of a T-second frame,

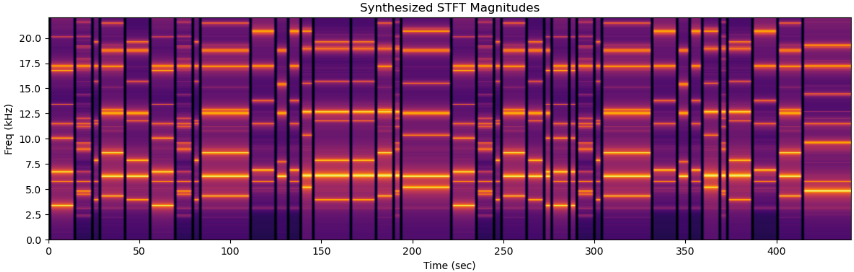
$$\phi(T) = \phi(0) + 2\pi f T$$

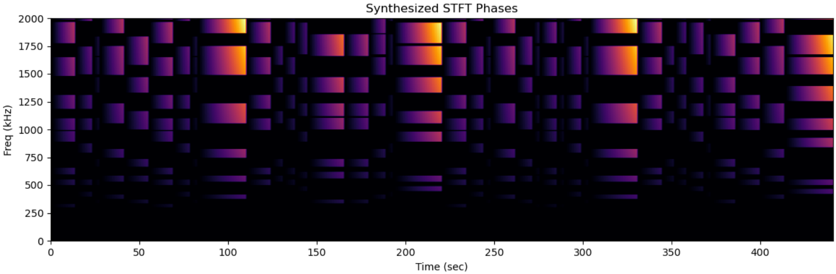
The phase vocoder

At each time t, for each of the DFT frequency bins k:

1. Decide whether $|X_t[k]|$ is one of the harmonics of a tone, or just noise.
2. If it's just noise, set $\phi_k(t)$ to a random number between 0 and 2π.
3. If it's a pure tone, set $\phi_k(t) = \phi_k(t-T) + 2\pi f_k T$, where f_k is the center frequency (kF_s/N), and T is the length of the frame (in seconds).

Result: synthesized magnitudes and phases





Outline

Summary

- **Semitones:** the number of semitones, s, separating two tones f_2 and f_1 is given by

$$s = 12\log_2\left(\frac{f_2}{f_1}\right)$$

- **The Harmonic Sieve algorithm:** choose the pitch with the best goodness, defined as

$$G(F_0) = \sum_{h=1}^{11} \sum_{k=0.95(hNF_0/F_s)}^{(1.05)hNF_0/F_s} |X[k]|$$

- **Phase Vocoder:**
  - If $|X[k]|$ is just noise, set $\phi_k(t)$ to a random number between 0 and 2π.
  - If $|X[k]|$ is a pure tone, set $\phi_k(t) = \phi_k(t-T) + 2\pi f_k T$, where f_k is the center frequency (kF_s/N), and T is the length of the frame (in seconds).