Eng. 100: Music Signal Processing
DSP Lecture 11
DSP topics / P3 help

Curiosity:  http://www.wired.com/2014/08/gyroscope-listening-hack
Curiosity:  https://www.youtube.com/watch?v=hSCOblXDCJc

Announcements:
• Course evaluations: email receipt to eecs-evals@umich.edu
• Final Exam: Thu. Dec. 17, 4-6 PM, 1500 EECS
Outline

• Part 1. P3 logistics
• Part 2. Finish BPM from last lecture
• Part 3. Filtering (by request)
• Part 4. Digital signals and quantization
• Part 5. Pitch and tempo shifting
• Part 6. Auto-tune
• Part 7. P3 help
Part 1. P3 logistics
P3 presentation schedule

● Each team should prepare 10 minutes of “talking” plus up to 5 minutes of integrated “live demonstration.” Total presentation should not exceed 15 minutes!

● 3-5 minutes of Q/A and transition

● Some of the live demonstration can be “pre-recorded” / playback, but some of it must be truly live.

● Tue Dec 8 lecture: 3-4 teams
  Thu Dec 10 lecture: 3-4 teams
  Preferences? ??

● First presentation should begin right at 10:40AM. Please come at 10:35AM to set up.
  Practice the audio/video in 1500 EECS beforehand

● All students must attend all presentations during Tue/Thu lectures.
P3 presentation/report items

Read P3 DSP specifications for required components, including:

- spectrum and/or spectrogram (Hz!)
- additive synthesis demo
- GUI screen shot, transcriber screen shot
- error-rate vs SNR plot...
Part 2. BPM
% bpm1_gen
% generate metronome tick signal to test bpm estimator

S = 8192;
bpm = 120;
bps = bpm / 60;  % beats per second
spb = 60 / bpm;  % seconds per beat
t0 = 0.01;  % each "tick" is this long
tt = 0:1/S:9;  % 9 seconds of ticking

f = 440;
%x = 0.9 * cos(2*pi*440*tt) .* (mod(tt, spb) < t0);  % tone
clf, subplot(211), rng(0)
x = randn(1,numel(tt)).* (mod(tt, spb) < t0) / 4.5;  % click via "envelope"
% sound(x, S)
% audiowrite('bpm1a.wav', x, S, 8)
% bpm1_find
% first try at beats-per-minute (bpm) estimator

[x S] = audioread('bpm1a.wav');
x = x'; % row vector
N = numel(x);

% a = real(ifft(abs(fft(x,2*N)).^2)); % autocorrelation
a = real(ifft(abs(fft(abs(x),2*N)).^2)); % why abs?

spacing = [0:(2*N-1)]/S; % why?
good = (spacing > 60/300 & spacing < 60/25); % min and max reasonable bpm
clf, subplot(211)
plot(spacing, a, 'b.-', spacing, 10*good, 'g.-', spacing, a .* good, 'r.-')
xlabel 'shift [s]', ylabel 'autocorrelation', axis([0 4 0 60])
 [~, index] = max(a .* good); % highest correlation for reasonable bpm range
disp(sprintf('estimated spb = %g, so bpm = %g', index/S, 60*S/index))
% ir_savefig -tight cw fig_bpm1b
BPM Summary

Is the preceding “metronome” signal periodic?

This small example illustrates several useful ideas.

• Noise blips
• Using modulo mod for repeating patterns
• Using logical operations like < to make binary signals.
• Constraining max to reasonable search range
• Looking for correlation between bursts of noisy signals using abs
• A few lines of Matlab code can do sophisticated DSP operations

Summary: (auto)correlation is quite widely useful
Part 3. Filtering
Music filtering example - spectra

Spectrum of original signal $x(t)$

Spectrum of filtered signal $z(t)$
% fig_filter1.m illustrate (low-pass) filtering
[x, S] = audioread('..\synth\cars.wav', [1 1e5]);
x = x(:,1); % mono
N = numel(x);
fx = fft(x);
cutoff_hz = [1000 4000];
cutoff_index = round(cutoff_hz/S*N)
fz = zeros(size(fx));
keep = [(1+cutoff_index(1)):(1+cutoff_index(2))];
fz(keep) = fx(keep);
z = real(ifft(fz));
Part 4. Digital signals and quantization
Continuous-time signals and discrete-time signals

Continuous-time = Analog signal

$$x(t) = \cos(2\pi ft)$$

Discrete–time (sampled) signal

$$x[n] = \cos\left(2\pi fn/S\right)$$
Digital signals and quantization

Discrete-time (sampled) signal

Quantization with 3 bits
Spectrum of quantized signal (3 bits)

Quantization with 3 bits

![Graph showing quantization with 3 bits](image)

Spectrum of $y[n]$

![Spectrum graph](image)
Spectrum of quantized signal (6 bits)

Quantization with 6 bits

original:  play  3 bit:  play  6 bit:  play  8 bit:  play
The default for .wav files is 16 bits. (Accepts 8, 16, 24, or 32.)
Part 5. Pitch and tempo shifting
Time scaling via sampling rate

Changing the sampling rate parameter: changes pitch \textit{and} tempo.

\texttt{sound(x, S)} \hspace{1cm} \texttt{sound(x, S * 2^{(6/12)})}
Tempo changes

How to play a recorded song faster or slower without changing pitch?

Phase vocoder

Basic idea:
- Use “short-time Fourier transform” (STFT) to make spectrogram (i.e., FFT of overlapping segments)
- Modify spectrogram using interpolation, being careful with phase
- Synthesize signal from modified spectrogram using inverse STFT (Inverse FFT via ifft of each segment, carefully combining.)

Example.

original:  play  3/4 speed:  play  3/2 speed:  play

Note: can combine phase vocoder with sampling rate parameter change
Spectrograms

Using \texttt{sound(x, 1.5*S)}

Using phase vocoder

original: 

play
Part 6. Auto-tune
Auto-tune demo

Popularized by Cher (!) in 1998 hit “Believe” [wiki]
http://www.mathworks.com/matlabcentral/fileexchange/26337-autotune-toy
tex/course/100-engin/demo/auto-tune-toy/AutoTuneToy.m

Recent example: https://www.youtube.com/watch?v=eq1FIvUHtt0
Part 7. P3 help

Questions?

References


