Music 421A
Winter 2011-2012
Homework #8
Phase Vocoder, Filter-Bank Summation, Additive Synthesis (Sinusoidal Modeling)
Due in two weeks

Theory Problems

1. **Digital Phase Vocoder**
   A precursor to the popular *sinusoidal modeling* technique was the *digital phase vocoder* (and analog vocoder before that).[1] Traditionally, the phase vocoder takes a single input of a human voice, splits the signal into several fixed frequency bands, computes the instantaneous-amplitude and instantaneous-frequency of the band (relative to the filter center frequency), then drives a bank of oscillators to resynthesize the sound. The analysis-resynthesis approach allows effects such as time stretching and pitch shift. A filter bank of bandpass filters is used to separate the input signal. We will look at the analysis portion of the phase vocoder.

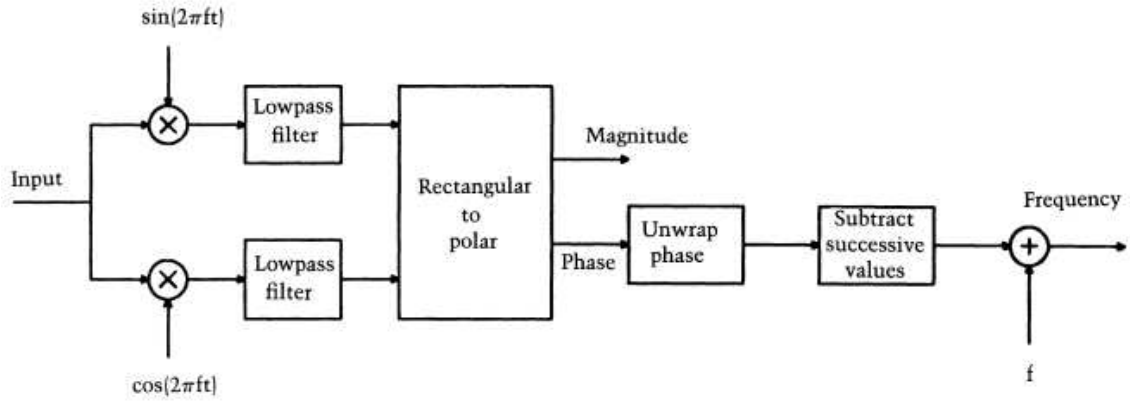

Figure 1: An individual Bandpass filter (From "The Phase Vocoder: A Tutorial" by Mark Dolson.

   (a) (2 pts) Given that we need the bandpass filters to be narrow enough to only capture one harmonic each, how many bandpass filters do we need in our filter bank to separate all of the harmonics of a periodic input signal at fundamental frequency 200 Hz, assuming that the sampling rate is 20 kHz?

[1] Search for "phase vocoder" on YouTube if you haven't heard one!

(b) (5 pts) As shown in Fig. 1, for a given band the signal is multiplied (modulated) by sine and cosine and lowpass filtered separately. What does this "ring modulation" do in the frequency domain for a 200 Hz sinusoid given a modulation frequency $f = 100$ Hz? In particular, what frequencies are present in the modulation output? (Consider the cosine- and sine-modulated branches both separately and together.)

(c) (2 pts) Given both the sine and cosine modulated signals, how are the time-varying magnitude and phase values computed?

(d) (1 pt) The output of the phase is first "unwrapped" and fed into the "subtract successive values" block. What is this approximating?

(e) (5 pts) When a large number of filters is required, FFT implementations of the digital phase vocoder using the STFT are more computationally efficient. Early FFT-based implementations, however, used windows longer than the FFT size. How is this possible? What benefit do you get from this? Do such windows obey weak or strong COLA? If so, which one?

2. (5 pts) Describe how to perform *cross-synthesis* using the phase vocoder.

3. (5 pts) Describe how to perform *time-scale modification* (TSM) using the phase vocoder.

Lab Assignment

1. (10 pts) **Getting Started with SDIF**

   (a) Download and run the Matlab program `make_sdif_file.m`[2] which shows how to make an `SDIF`[3] file in Matlab. Change the filename of the output file appropriately. It uses IRCAM's[4] SDIF Extensions for Matlab[5] (which should already be installed at CCRMA) to write a Matlab cell array to an SDIF file. (Later in this homework you will write SDIF files containing the results of a sinusoidal analysis of an input sound. For now, the parameters in the SDIF file are just some made-up numbers so that you can see how SDIF works.)

   (b) Use the Unix command-line utility `spew-sdif` to print the contents of the SDIF file. This should be installed in `/usr/ccrma/bin`, which should be part of your Unix path by default. If it is not there, you can run `/usr/ccrma/bin/spew-sdif filename`.

[2] `http://ccrma.stanford.edu/~jos/sasp/hw/make_sdif_file.m`
[3] http://cnmat.berkeley.edu/library/sdif/
[4] http://www.ircam.fr
[5] http://recherche.ircam.fr/equipes/analyse-synthese/sdif/download/sdif-matlab/

(c) Download the Matlab program `additive_synth.m`[6] and use it to synthesize the SDIF file you just created. Listen to the resulting sound.

(d) Changing only the index numbers (i.e., the integers in the first columns of the matrices `frame1`, `frame2`, etc), modify `make_sdif_file.m` so that it produces an SDIF file that sounds noticeably different when you synthesize it.

Full credit is given for evidence (code, spectrogram, and/or sound files) that all steps have been completed.

2. (20 pts) Write a matlab program to perform sinusoidal analysis on an input sound and write the result to an SDIF file. You should use the `findpeaks` function from HW#4 or elsewhere. The parameters of interest are the amplitudes and frequencies of the sinusoidal components. (Don't worry about phase.) When matching up spectral peaks in adjacent frames, choose the solution that minimizes frequency deviation from one frame to the next.

   Your program should take the following arguments:

   - Filename for resulting SDIF file
   - Input samples
   - Sampling rate f_s of input samples
   - Analysis frame rate f_R (FFT hop size $R = \text{floor}(f_s/f_R)$)
   - Percentage overlap of analysis frames
   - Window function
   - Any other analysis parameters appropriate to your analysis technique

   Download the sound file `wrenpn1.wav`[7] which contains the sound of a bird chirping with noise embedded. Test your analysis program on this sound file using the following parameters:

   - Analysis frame rate $f_R = 200$ Hz
   - 50% overlap of analysis frames ($R \approx M/2$, where M is the window length)
   - Window each frame by Hann window before analysis (zero-phase is not needed).
   - Use the number of peaks which gives you the best results.

   Use the `additive_synth.m` oscillator-bank additive-synthesis program to technique to reconstruct the birdsong based on your sinusoidal model, stripping out the noise in the process. (The signal to noise ratio is approximately 60 dB). Use the `'ignore_phase'` interpolation type argument to get linear interpolation on both the amplitudes and frequencies throughout.

   Turn in your analysis code and your final denoised soundfile. You can email them to the TA, or create a temporary webpage for the TA to view.

[6] `http://ccrma.stanford.edu/~jos/sasp/hw/additive_synth.m`
[7] `http://ccrma.stanford.edu/~jos/sasp/hw/wrenpn1.wav`

(a) (5 pts) Plot the following spectrograms using Matlab's `specgram` function:

   i. Original birdsong.

   ii. Resynthesized birdsong.

(b) (5 pts) At the time half way through, plot the following spectral slices overlaid on a dB scale (i.e., just plot the spectral magnitude at that time):

   i. Original birdsong.

   ii. Resynthesized birdsong.

3. (10 pts) For this problem you will write two general-purpose programs that transform sinusoidal models stored in SDIF files. Each program should read in an input SDIF file and write the result to an output SDIF file. You should then run the output SDIF file through the additive synthesizer to hear the result.

   (a) (5 pts) Frequency-scale modification: Decrease the pitch of the birdsong by a factor of 4 without changing duration, creating a "bigger bird" sound. Plot the spectrogram of the result.

   (b) (5 pts) Time-scale modification: Stretch the birdsong in time by a factor of 2, keeping the frequency excursions unchanged, and plot the spectrogram of the result. (Try some more extreme slow-down factors, just for fun).

   Extra credit: Instead of having the frequency and time scale factors be constants, allow them to be functions of time. For example, you should be able to decrease the pitch of the birdsong by a factor of 2 initially, then have it keep getting lower over the course of the SDIF file, to an ending pitch factor of 6.