Timbral Distortion in Inverse FFT Synthesis

Mark Zadel

Introduction

Inverse FFT synthesis (FFT\(^{-1}\)) is a computationally efficient technique for performing additive synthesis [1]. Instead of summing partials explicitly as in the oscillator bank method, synthesis is performed by manually generating a sequence of short term spectra and performing an inverse short time Fourier transform to generate the time domain signal.

A number of optimizations are made to maximize the efficiency of this process, and the frequency and amplitude of each partial are assumed to be constant over each frame. While this simplifies the generation of spectra, it can lead to distortion in the synthesized signal. Partial with changing frequency are represented as an overlapped set of short-term sine waves of constant frequency, potentially leading to amplitude modulation in the overlap region and timbral distortion of the sound. An example can be seen in figure 1.

Cubic phase unwrapping [2] is a technique for smoothly interpolating the phase of partials with changing frequency. The phases and frequencies for each partial are specified at the centre of each frame. The instantaneous phase is computed as a piecewise cubic polynomial which interpolates the specified phases and has the specified frequencies (slopes) at the breakpoints. The breakpoint phases are adjusted by integer multiples of 2\(\pi\) in order ensure maximally smooth phase evolution.

This paper presents an implementation of both FFT\(^{-1}\) and the oscillator bank method using cubic phase unwrapping. The code synthesizes example sounds using both methods in order to illustrate the timbral distortion described above. The implementation’s workings and limitations are described, and ideas for integrating the two synthesis methods are given.

Figure 1: A demonstration of the distortion that can produced by FFT$^{-1}$ when synthesizing changing frequencies: (a) the original partial with frequency sweeping upward; (b) the constant-frequency windowed frames representing the partial in FFT$^{-1}$; (c) the overlap/add result, which exhibits distortion.

Overview, running the examples

The corresponding MATLAB code implements both FFT$^{-1}$ and the oscillator bank method for additive synthesis. A simple set of examples are run which illustrate the timbral distortion that can occur in FFT$^{-1}$. The same examples are synthesized using the oscillator bank method, which does not suffer from this kind of distortion.

The examples are run by executing the zadel605finalproject MATLAB function in the zadel605finalproject.m file. All of the required sub-functions appear in this file. A single partial with sweeping frequency is synthesized using both synthesis methods, using three different frame sizes (128, 512, 4096). These are output in six wav files in the current directory. The ones synthesized using FFT$^{-1}$ exhibit timbral distortion described above due to the partial’s quickly changing frequency. The distortion is worse for larger frame sizes since the changing frequency is encoded more coarsely than for smaller frame sizes. The oscillator bank/cubic phase unwrapping synthesis does not exhibit any of this distortion.

Two other wav files are produced, one per synthesis method, which give examples of multiple partials with changing frequencies and amplitudes. These demonstrate that the implementation works for multiple partials.
Implementation

The FFT\(^{-1}\) code is contained in the methods `generatesequenceofspectra`, `synthesizespecseq` and `overlapadd`. `generatespecseq` generates a sequence of spectral domain frames from input sequences of frequencies and amplitudes for each partial, and a list of their initial phases. The phases for each partial at the beginning of each frame are computed from the initial phases and the sequence of frequencies. The FFT of one Blackman-Harris window is computed and stored. Blackman-Harris is used for efficiency since its Fourier transform has few significant values, the rest of which are ignored [3]. This is oversampled using linear interpolation to increase the frequency resolution. The FFT of each partial is computed for each frame by shifting the window and adjusting for the appropriate phase and amplitude. These are summed to generate the short term spectrum of the signal for each frame. `synthesizespecseq` generates a time domain signal from a sequence of spectra. The inverse Fourier transform is computed for each frame. The Blackman-Harris window used in `generatesequenceofspectra` is divided out of each frame, and the result is windowed with a triangular window. Overlap/add is performed on the sequence of time domain frames to produce the final signal. `overlapadd` combines the sequence of time-domain frames into the final signal. The time domain frames are arranged in a large sparse matrix such that one frame appears in each column, and they are all positioned appropriately for overlapping. The matrix rows are then summed to produce the final signal.

The oscillator bank/cubic phase unwrapping code is contained in `oscillatorbank`, `linearampinterpolation`, and `cubicphaseinterpolation`. `oscillatorbank` generates the final time domain signal from a sequence of frequencies, amplitudes and phases. The instantaneous amplitude of each partial is computed at each time in the signal, as are the instantaneous phases. The cosine of each phase value is computed and multiplied by the corresponding amplitude value. These instantaneous partial values are added to produce the final time domain signal. `linearampinterpolation` computes the instantaneous amplitude values from a sequence of amplitudes using linear interpolation. The amplitudes are assumed to be measured at the centre of each frame. `cubicphaseinterpolation` computes the instantaneous phases for each partial using cubic phase unwrapping. The input phases are assumed to be measured at the centre of each frame.

Both of these methods assume a constant number of partials over the entire signal. The frame sizes are assumed to be powers of two, and the frames are assumed to overlap by a half frame. Some suggestions for computational efficiency in FFT\(^{-1}\) are not implemented; in particular the entire window FFT image is used, including the insignificant values. The MATLAB code vectorizes operations as

```matlab
3
```
Much as possible, and uses techniques from [4] [5].

Discussion

Both synthesis methods were implemented successfully, and the examples illustrate the timbral distortion that can occur using $\text{FFT}^{-1}$. The $\text{FFT}^{-1}$ implementation suffers from another kind of distortion, however, stemming from the linearly interpolated window FFT used to generate the sequence of spectra. This interpolation is required since the low frame size leads to large frequency bins, and each peak must be placed more finely within each bin. The oversampling allows the peak to be positioned more precisely. Unfortunately, the linear interpolation produces inaccurate window FFT images for frequencies in the middle of each bin, and consequently produces distortion. It is not possible to simply use a high resolution window FFT since the shape depends on the size of the FFT, and thus the image has to be computed using the actual frame size. A potential solution to this would be to take Fourier transforms of windowed signals of frequencies throughout a frequency bin, and then using the appropriate window for each partial for each frame.

As mentioned above, the $\text{FFT}^{-1}$ synthesized examples exhibit timbral distortion due to the constant frequency assumed over each frame, and the oscillator bank/cubic phase unwrapping synthesis does not exhibit this distortion. This suggests that the shortcomings of $\text{FFT}^{-1}$ could be addressed by cubically interpolating the phase of each partial over each frame. The challenge would then be to determine a priori an FFT image based on this changing frequency. If a closed form for this window image can be derived, this could be used to eliminate the timbral distortion seen here.

Conclusion

An implementation of $\text{FFT}^{-1}$ and cubic phase unwrapping was described. The code generates examples which illustrate the timbral distortion that can result from $\text{FFT}^{-1}$. The details of the implementation were described. Combining $\text{FFT}^{-1}$ with cubic phase unwrapping could provide the efficiency benefits of the former while eliminating the potential for timbral artifacts.

References


