Using Noise Substitution for Backwards-Compatible Audio Codec Improvement

Colin Raffel

AES 129th Convention
San Francisco, CA

February 16, 2011

Outline

- Introduction and Motivation
- Coding Error
- Analysis
- Synthesis
- Example: “row-mp3”

Introduction and Motivation

- Problem: Many widely used audio codecs are out of date compared to the state-of-the-art because they were not made to be improved upon in a backwards-compatible way

Introduction and Motivation

- Problem: Many widely used audio codecs are out of date compared to the state-of-the-art because they were not made to be improved upon in a backwards-compatible way
- Observation: Adding the coding process's residual, or coding error, back into the audio file will result in a “lossless” audio quality

Introduction and Motivation

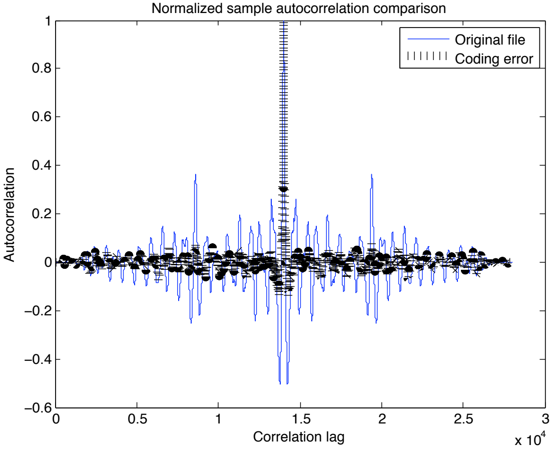
- Problem: Many widely used audio codecs are out of date compared to the state-of-the-art because they were not made to be improved upon in a backwards-compatible way
- Observation: Adding the coding process's residual, or coding error, back into the audio file will result in a "lossless" audio quality
- Technique: Find a way to represent the coding error with a small amount of data and include the information in the coded audio file

Introduction and Motivation

- Problem: Many widely used audio codecs are out of date compared to the state-of-the-art because they were not made to be improved upon in a backwards-compatible way
- Observation: Adding the coding process's residual, or coding error, back into the audio file will result in a "lossless" audio quality
- Technique: Find a way to represent the coding error with a small amount of data and include the information in the coded audio file
- Proposal: Store frame-by-frame, per-critical-band residual levels in the audio codec's metadata and re-synthesize the coding error as colored noise when decoding

Coding Error

- Achieving lower data rates requires some information loss
- We can define coding error as $(\textit{original audio}) - (\textit{coded audio})$
- Tends to be noisy ▷
- Modeling as colored noise is cheap



Residual Analysis: Spectral Flux

- Idea: Model only the non-stationary component of the error
- Simple method: Spectral flux, defined as

$$\mathrm{SF}(n) = \sqrt{\sum_{k=0}^{N-1} (|X[n,k]| - |X[n-1,k]|)^2}$$

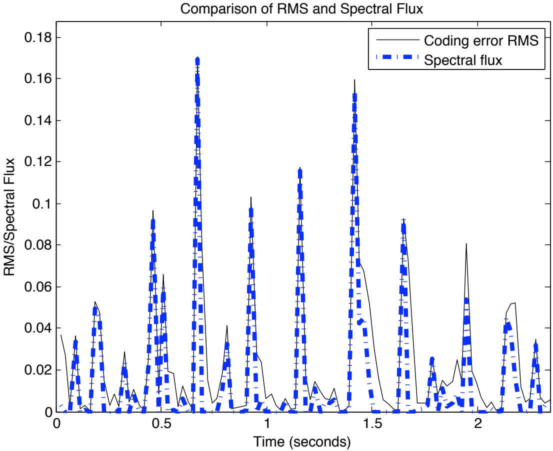
- Stationary signal components get subtracted out
- Roughly speaking,

$$\mathrm{SF}(n) \propto \mathrm{RMS}(x[n])$$

- Full proof is in the paper
- Proportionality only holds for Gaussian noise and non-overlapping rectangular windows

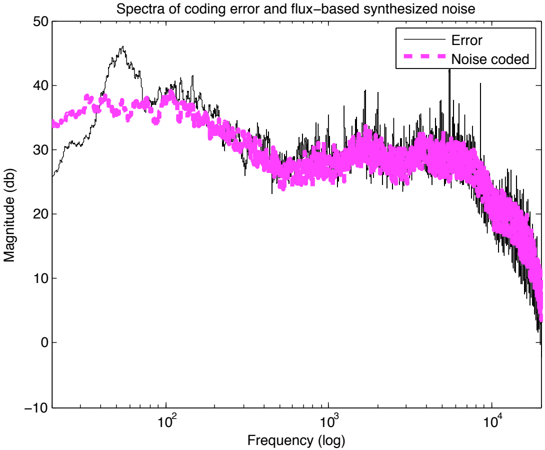
Residual Analysis: Spectral Flux

- Coding error does not satisfy proportionality criterion
- The proportionality still roughly holds in practice



Residual Analysis: Spectral Flux

- To determine coloring, evaluate the flux on a per-band basis
- Band levels tended to change too rapidly from frame-to-frame
- However, RMS proportionality holds in practice and makes this technique useful ▷



Residual Analysis: Smoothed Cepstrum

- Obtain spectral envelope by windowing the real cepstrum and taking the DFT

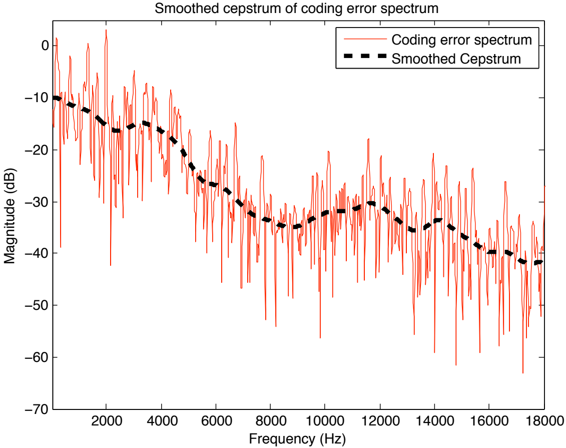
$$C[n] = \Re\left(\frac{1}{N}\sum_{k=0}^{N-1}\log(|X(k)|)e^{j2\pi nk/N}\right)$$

$$E[k] = \Re\left(\sum_{n=0}^{N-1}w[n]C[n]e^{-j2\pi nk/N}\right)$$

- Works well for relatively peak-free spectra
- Per-band level can be found by averaging over bins in band

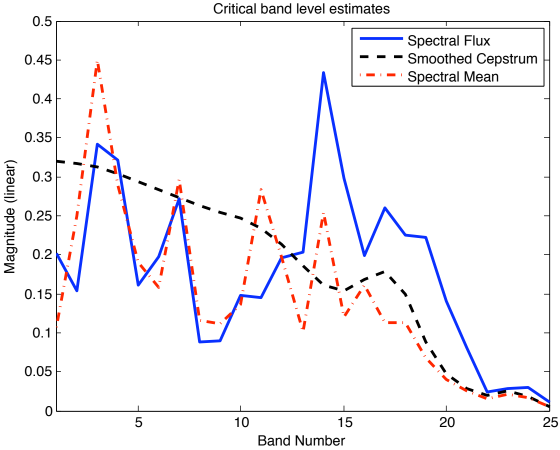
Residual Analysis: Smoothed Cepstrum

- Generally results in band levels which are “smooth” from band to band and frame to frame ▷



Residual Analysis: Comparison

- Flux is analytically “clean”, but varies rapidly because it is intentionally uncorrelated
- Smoothed cepstrum provides a reasonable estimate which is smoother in time and band



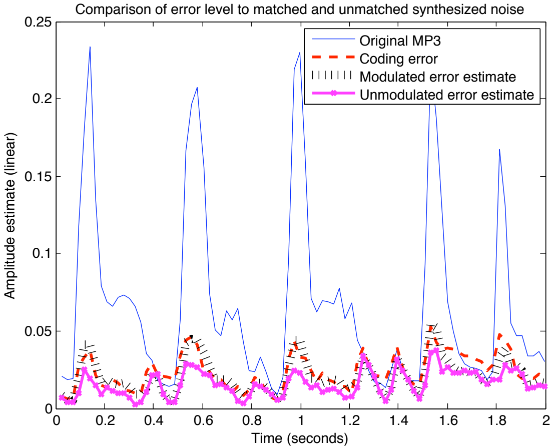
Residual Synthesis

- Generate coding error representation by applying critical band envelopes to a random spectra ▷
- Envelope differences from frame-to-frame cause coloring discontinuities
  - We can generate any amount of colored noise by generating a larger spectrum
  - So, create additional noise per-frame and crossfade
- Transients in the residual result in frames of noise in the error representation
  - Traditional methods for detecting and representing transients are not effective
  - The coded audio and coding error's envelopes are similar
  - We can modulate residual representation with the coded audio's envelope

Residual Synthesis

- We can parametrize the amount of envelope modulation by ▷

$$y[n] = ((1-\alpha) + \alpha L[n]))\, x[n]$$



Implementation: “row-mp3”

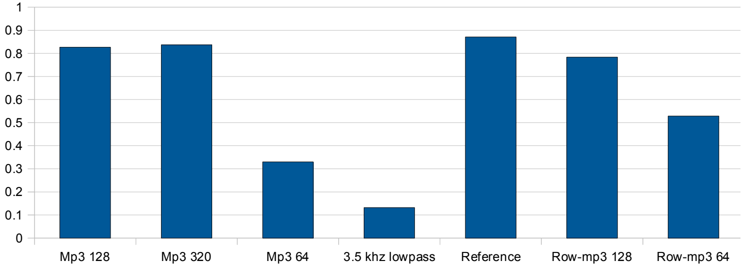
- The MP3 codec is highly pervasive but somewhat out-of-date
- To allow backwards-compatibility, we can store information in the ID3 (metadata) tag
- “row-mp3”-aware decoders can use the information, while others will simply ignore it
- Including per-frame critical band levels results in a relatively small data overhead
  - For example, with a 23.2 ms frame size and 8-bit quantized band level values we have

$$(.0232) * (8) * (25) = 8.6 \text{ kbit/s/channel}$$

- Data overhead can be reduced by using different quantization schemes or compression such as Huffman coding

Implementation: “row-mp3”

- Created a simple MUSHRA-like web-based test to determine codec’s effectiveness
- row-mp3 files used spectral flux method with no envelope modulation
- 60 subjects tended to rate the row-mp3 version about 150% better for low MP3 bit rates
- Further, more controlled testing with all error analysis and synthesis methods is needed



Conclusions

- Audio coding error can be effectively modeled as colored noise
- Flux provided a “theoretically-sound” coloring estimate
- Cepstral smoothing works better in practice
- Synthesis by scaling random spectra
- Cross-fading and interpolation prevented coloring discontinuities
- “Level-modulated” error estimate helped prevent smeared transients
- row-mp3 codec and accompanying listening tests suggest feasibility

Future work

- Investigating the optimal number and spacing of bands
- Testing the effectiveness of other analysis techniques
- Evaluating different methods for dealing with transients
- Applying similar techniques to spectral modeling and other processes with residual
- Implementing inclusion schemes in other audio codecs
- Generating residual levels solely from the coded audio (as a sound enhancement)

Acknowledgements

- Jieun **O**h and Isaac **W**ang for creating the “row-mp3” codec
- Prof. Marina Bosi for her instruction in the field of audio coding
- Prof. Julius Smith for helpful advice and discussion on various topics

Sound examples and code

http://ccrma.stanford.edu/~craffel/software/noise/

http://ccrma.stanford.edu/~craffel/software/rowmp3/