Accelerated Gammatones

Coen D. Needell

2020-05-12T14:37:01-05:00

Introduction

Last winter, I worked on a personal project I call *ongaku* (from the Japanese for 'music'). This was an attempt to use manifold learning to create a metric space for music. The preprocessing relied heavily on a method called (Valero and Alias 2012). This method was intended to replace Mel Frequency Cepstral Coefficients. Where Mel Frequency is a logarithmic transformation of sound frequency, in an attempt to simulate human perception of sound. The most common transformation for Mels is:

$$m = 2595 \log_{10} 1 + \frac{f}{700}$$

This simple approach works very well, but it has its basis in human *perception* which is a tricky thing. There's no reason to believe that the brain makes judgments based on human perception. It's very possible that the brain takes in data, and produces two perceptions, one of the sound itself, and the other as the semiotic interpretation of that sound. Enter the Gammatone Cepstral Coefficients. This is a simplification of the process that sound signals undergo when being transferred through the cochlear nerve, the nerve that transfers data between the ear and the brain. So the theory goes, this will allow a machine learning algorithm to work with data that better simulates how the brain receives sound signals, rather than how the mind perceives them. Research has shown that GFCCs outperform MFCCs in machine learning tasks (Liu 2018). The gammatone transformation looks like:

$$g_q(t) = t^4 e^{-54\pi t + j20\pi t} u(t)$$

I have been able to find an implementation of the gammatone transformation in python, but it's slow, and is a port of a MATLAB plugin. This package is intended to mimic that package's structure, but instead working natively in OpenCL for speed. This is problematic for a number of reasons. First, the MATLAB plugin has been shown to be inefficiently implemented (Ma, n.d.a)

(Ma, n.d.b). Secondly, for my purposes, gammatones need to be processed on long files, 2-10 minutes in length, and even Dr. Ma's implementation runs serially. As such, I believe that an inherently parallel version of this would be a benefit, not only to me, but to the scientific computing community at large. As such, this project will be released on pypi as `gammatone`. There does exist a github project called `gammatone` which is the implementation that ongaku currently uses, but it is not avaliable on pypi at the time of writing, so there will be no conflicts in the long run.

Implementation

The available implementations of gammatone filters are serial. However, there have been a number of attempts to parallelize IIR (Infinite Impulse Response) filters in the past(Anwar and Sung, n.d.) (Belloch et al. 2014). The general consensus is that this is rarely better than a serial implementation, and the implementation of a new filter needs to be made bespoke for that filter. After analyzing the implementation methods, I decided that (for now,) this is beyond the scope of this project. The fourth order gammatone filter is implemented like:

$$erb(x) = 24.7*(4.37\times10^{-3}+1)\delta = e^{\frac{2\pi}{f}erb(f_c)\times1.019}q_t = \cos\frac{2\pi}{f}f_c+i\sin\frac{2\pi}{f}f_cg = \frac{\left(\frac{2\pi}{cf}erb(f_c)\times1.019\right)^4}{3}y_t = q_tx_t+$$

Where f is the sampling frequency of the signal, and f_c is the target frequency to test membrane resonance against. The basilar membrane displacement (B_t) and Hilbert envelope (H_i) are defined with simple transformations. The cochleagram uses the Hilbert envelope to construct an image.

$$B_t = gy_tq_tH_t = g\sqrt{y_t^2}$$

Another option is to use a prefix sum (Blelloch, n.d.). The problem with this is that while a single prefix sum can be exact to the first order of a filter, most implementations of gammatone cochleagrams use a fourth order filter. However, most implementations are made for medical applications, and I am a social scientist. Implementing a first order gammatone filter can, unlike higher order filters, can be reduced to a one-dimensional prefix sum operation (Blelloch, n.d.). A prefix sum is generally defined with an operator $a \oplus b$ such that:

$$y_t = \bigoplus_{\forall t} x_ty_t = x_0 \oplus x_1 \oplus \cdots \oplus x_t$$

For a first order gammatone filter we can define:

$$a \oplus b = \alpha\delta a + b$$

Then implementing the gammatone filter is a matter of finding the best α. Based on the original description of gammatone filters, we'll set $\alpha = -.67$. It must be negative so that the filter can resonate with the input signal (Shen, Sivakumar, and Richards 2014).

Results

In comparison with the detly implementation, testing across all of the songs in the album Traffic and Weather by Fountains of Wayne (Wayne 2007). These songs are 2 minutes, 58 seconds on average. On my laptop, which has an Intel Core i7-7700HQ CPU @ 2.8GHzx8 processor and a GeForce GTX 1080 Max-Q GPU, the detly implementation (serial) takes 7.5 seconds to create a 16-level Cochleagram (16 gammatone filtrations) for a single song on average. The GPU implementation takes 1.75 seconds for a single song on average. This is a little more than a 4x speedup.

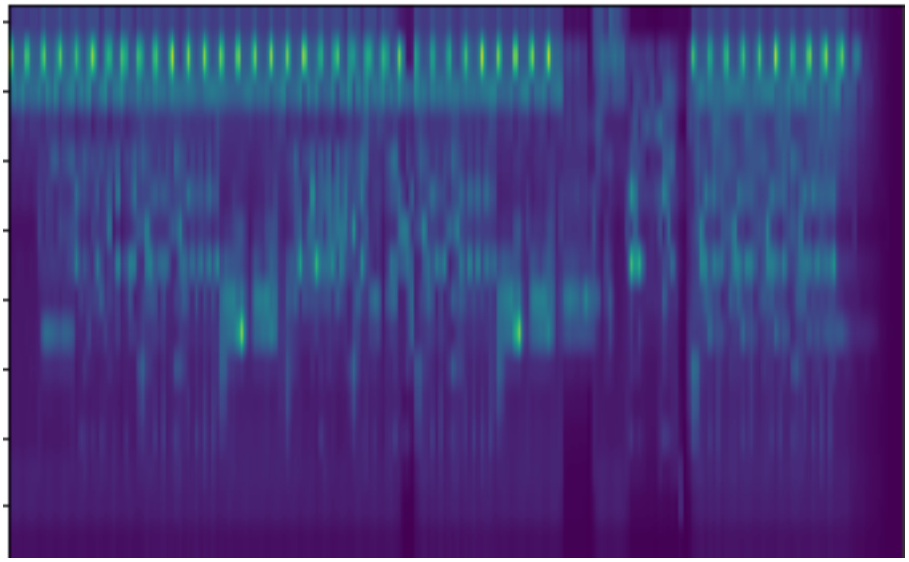

Figure 1: Figure 1: The cochleagram created by the serial implementation

Figures 1 and 2 show the old and new implementation's cochleagrams. Notice that the new implementation has a "squishy" quality to it. Which is expected since it's a much less complicated implementation. Despite the lack of clear definition, a machine learning technique should still be able to identify aural features.

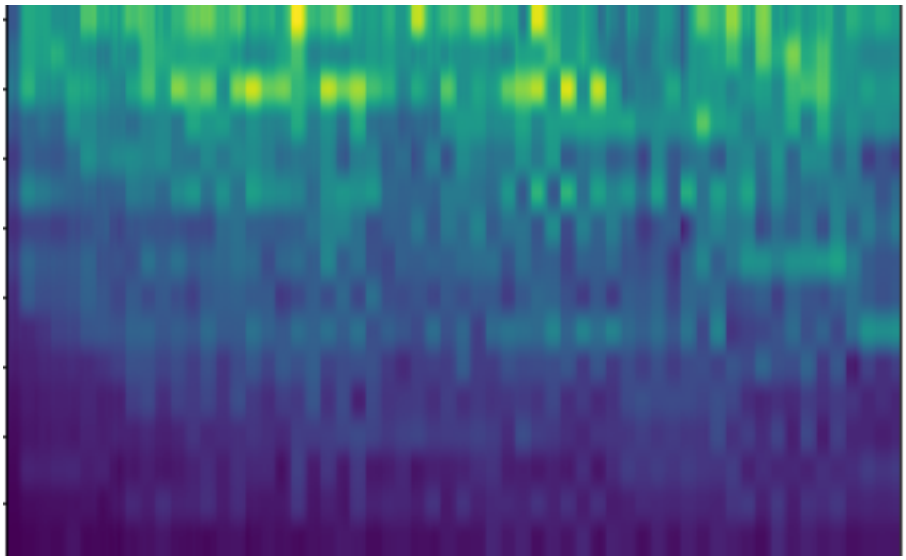

Figure 2: Figure 2: The cochleagram created by the GPU implementation

Future

While creating a higher order implementation of the gammatone filter and therefore the cochleagram is outside the scope of this project, I believe it can be done given more time and outside research. A possible pathway is to create a "horizontal" kernel, that runs multiple fourth order gammatone filters at once. Another approach is to convert the signal to frequency space, and then apply multiple frequency space filters to the signal in a cascading manner (Holdsworth, Patterson, and Nimmo-Smith, n.d.). Even though this project has resulted in a small step in the right direction, it could ultimately unlock the door for more accessible social science research on audio content.

References

Anwar, Sajid, and Wonyong Sung. n.d. "Digital Signal Processing Filtering with GPU," 2.

Belloch, Jose A., Balazs Bank, Lauri Savioja, Alberto Gonzalez, and Vesa Valimaki. 2014. "Multi-Channel IIR Filtering of Audio Signals Using a GPU." In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6692–6. Florence, Italy: IEEE. https://doi.org/10.1109/ICASSP.2014.6854895.

Blelloch, Guy E. n.d. "Prefix Sums and Their Applications," 26.

Holdsworth, John, Roy Patterson, and Ian Nimmo-Smith. n.d. "Implementing a

GammaTone Filter Bank," 5.

Liu, Gabrielle K. 2018. "Evaluating Gammatone Frequency Cepstral Coefficients with Neural Networks for Emotion Recognition from Speech." *arXiv:1806.09010 [Cs, Eess]*, June. http://arxiv.org/abs/1806.09010.

Ma, Ning. n.d.a. "An Efficient Implementation of Gammatone Filters." https://staffwww.dcs.shef.ac.uk/people/N.Ma/resources/gammatone/.

———. n.d.b. "Cochleagram Representaion of Sound." https://staffwww.dcs.shef.ac.uk/people/N.Ma/resources/r

Shen, Yi, Rajeswari Sivakumar, and Virginia M. Richards. 2014. "Rapid Estimation of High-Parameter Auditory-Filter Shapes." *The Journal of the Acoustical Society of America* 136 (4): 1857–68. https://doi.org/10.1121/1.4894785.

Valero, X., and F. Alias. 2012. "Gammatone Cepstral Coefficients: Biologically Inspired Features for Non-Speech Audio Classification." *IEEE Transactions on Multimedia* 14 (6): 1684–9. https://doi.org/10.1109/TMM.2012.2199972.

Wayne, Fountains of. 2007. "Traffic and Weather." Virgin.