

This short paper by Klapuri described two extensions of previous algorithms that attempt to automatically transcribe music from a given audio stream. The first extension applied to the onset detector, removing non-harmonic noise components from percussion instruments from the audio. The second extension was to a multi-pitch estimation algorithm that was capable of detecting polyphonic onsets. The extension involved an iterative method to calculate the number of polyphonic voices by acoustic analysis.

The noise suppression algorithm took as the power spectrum $X(k)$ of the input audio and transformed it into the spectrum function $Y(k)$ where:

$$Y(k) = \left(1 + \left(\frac{k_1 - k_0}{\sum_{k=k_0}^{k_1} X(k)^{1/3}} \right)^3 \times X(k) \right)$$

In this function k_0 and k_1 represent frequency bounds of 50 Hz to 6 kHz and the function represents an averaging scheme using the cube root. This averaged form has a noise component $M(k)$ which is then estimated using a hamming window around the frequency f corresponding to k . Finally the noise is suppressed by simply subtracting $M(k)$ from $Y(k)$ and bounding the result by zero on the lower end.

This paper was very light on reasoning behind the signal processing. Part of it may be my lack of signal processing experience, but while the initial averaging step to produce $Y(k)$ seemed at least somewhat intuitive, I don't see how the application of the Hamming window (which should merely reduce the power spectra of regions at the edges of the window) produces the noise estimation. The window size is a linear function of the frequency, so the higher frequencies get wider windows, but I don't see how this correlates to noise in the signal.

The next extension was an iterative method to track the number of concurrent voices. The author neglected to even briefly discuss the previous algorithm that he extends, but I extrapolated partially from the text. The algorithm iteratively identified the most likely pitch, and its associated noise spectrum. From there the iterative step would presumably be removing the particular frequency peak and repeating until only noise remains. I can't be certain this was what was intended but intuitively this must be the only way this algorithm can work as described. The weaknesses I can see with this approach are when instrument onsets coincide at the same pitch, with the timbre differences being lost in the interplay of the the two instruments' harmonics. The algorithm may be apt to simply discard one instrument's information. Of course, this would only be a larger scale problem if every note coincided for both instruments, so perhaps this is not such a bad algorithm after all.

At the end, the author discussed preliminary techniques for stream separation but did not go into depth. This problem seemed an even more exception fraught one than simple polyphonic onset identification, but definitely an area worth investigation. The proposed scheme involved building an information vector for each instrument in the polyphonic sample, that described the instrument's timbre from a signal processing view. A clustering algorithm could then match each separated monophonic stream against the different instrument patterns which could then be used to resynthesize the original input using MIDI or other instruments mapped from each instrument sample. I don't think the problem will be as easy as it is made out to be, there are too many variables in play when it comes to timbre, especially when families of different instruments share particular patterns and harmonics.