MULTIPLE FUNDAMENTAL FREQUENCY ESTIMATION

Michael Groble
mg2467@columbia.edu

ABSTRACT

This paper describes a submission to the 2008 Music Information Retrieval Evaluation eXchange in the Multiple Fundamental Frequency Estimation Task.

1 OVERVIEW

This is a new system which began development in the Spring of 2008. The initial focus was to perform real-time MIDI transcription of guitar music. The system has been generalized slightly for submission to MIREX, but the emphasis remains on relatively simple feature calculations and fast estimation.

2 FEATURES

Various features are computed each 441 frames or 10 ms of audio. The base feature vector x_f computed for frame f is the first 180 frequency bins of the magnitudes of the short-term Fourier transform over a 2048 sample frame. In addition to this base feature vector, the following features are computed for each frame

$$\begin{aligned} dx_f &= x_f - x_{f-1} \\ n_f &= \left\|x_f\right\| \\ \delta_f &= \left\|dx_f\right\| \\ s_f &= \frac{x_f \cdot x_{f-1}}{x_{f-1} \cdot x_{f-1}} \\ r_f &= \frac{\delta_f}{n_f} \\ p_f &= r_f^2 - \left(\frac{1}{3}\sum_{i=1}^{3} r_{f-i}\right)\left(\frac{1}{3}\sum_{i=1}^{3} r_{f+i}\right) \end{aligned}$$

dx_f is the vector difference between the current feature vector and the previous one. n_f is the norm of the feature vector. δ_f is the norm of the difference between the current and previous feature vectors. s_f is a scaling parameter measuring the projection of the current feature vector as compared to the previous feature vector. r_f is a measure of the relative difference in the feature vector norms. Finally, p_f is a peak indicator which compares the current frame value of r_f with the average of preceding and following values. This peak indicator is similar to the Nonlinear Energy Operator described in [2].

3 NOTE PREDICTION

3.1 Frequency Prediction

As in [3], note prediction is treated as a classification problem, instead of a problem of estimating the fundamental frequencies. The underlying system estimates the MIDI note numbers detected in each frame of audio. To meet submission guidelines, the MIDI note numbers are converted to frequencies corresponding to perfectly tuned notes.

Each pitch has a model associated with it computed by scaled averages of audio samples. Frames are scored against these models to determine which pitches are present in the frame.

With scaled averaging, the model for a particular pitch is generated from a large number of sample frames taken from an instrument playing that note. The training data contains samples at multiple note attack levels and multiple distances between the note onset and the frame onset. The model for a particular pitch is computed by normalizing each individual feature vector at that pitch to have a unit height, averaging the unit height vectors together, and normalizing the resulting average vector to have unit length. More formally, the model vector m_p for a specific pitch p is computed from samples x_i with pitch labels y_i as follows

$$\begin{aligned} a_p &= \sum_{j:y_j=p} \frac{x_j}{\left\|x_j\right\|_\infty} \\ m_p &= \frac{a_p}{\left\|a_p\right\|} \end{aligned}$$

In the simplest case, for a single note the predicted pitch p_{est} for a feature vector x is then determined as the one satisfying

$$p_{est} = \arg\min_p \left\| \frac{x}{\|x\|} - m_p \right\|_1 \quad (1)$$

For multiple notes, a frame profile is built up note-by-note iteratively. This is similar to [1], where the single best scoring note is determined from the feature vector, then the feature vector is modified to "remove" the effects of that

pitch. The remainder is then rescored until the desired number of notes have been predicted. With the non-linear scoring metric in (1), a slightly different approach is required. In particular, a base profile $base^i$ at iteration i stores the feature vector representing the pitches which have already been detected for the current frame. Pitches for iteration i are then scored as follows with the initial base set to a zero vector $base^0 = 0$.

$$\begin{aligned} score_p^i &= \left\| \frac{x}{\|x\|} - \frac{m_p + base^{i-1}}{\left\|m_p + base^{i-1}\right\|} \right\|_1 \\ p_{est}^i &= \arg\min_p score_p^i \\ base^i &= base^{i-1} + m_{p_{est}^i} \end{aligned}$$

Additionally, each iteration does not need to include models of just single pitches. In *combined* scoring, models are also developed from frames where pairs of notes are sounding together. Each iteration starts by computing a ranked list of pitch scores $score^i$. Note combinations are generated by combining highly ranked pitches and scoring the combination. If the best combination scores better than the best single-pitch score, the combined pitches are added to the estimated pitches and the corresponding combined feature vector added to the base.

Iteration stops when the best score is no better than a score against a sentinel zero model vector

$$m_\emptyset = 0$$

Choosing this as a cutoff is effectively the same as choosing the cutoff to be the case when the remainder of the frame is closer to the zero vector than it is to all other pitches in the model. Analysis of the single note model and combined note model methods on a guitar music corpus showed the need to introduce a scaling factor σ for the sentinel to be able to control the trade off between precision in recall. Low recall is sometimes an indication that the termination criteria is too strict. In particular, deletion errors are high because the algorithm is deciding there are no more notes to estimate when there really are. The scale factor σ is used to multiply the sentinel score before comparing with the other scores. Higher values for σ therefore means a more lax stopping criteria (since the best score is the lowest score). Higher values therefore minimize deletes and increase recall (at the possible consequence of more substitutions) while lower values generally reduce inserts, increase precision and decrease recall.

3.2 Note Onset Prediction

Analysis of the guitar music corpus shows that p_f thresholds provide high correlations with frames having note onsets or terminations in them. Based on this information, notes are estimated for contiguous blocks of frames.

Specifically, the raw feature set is computed every 10 ms and the p_f threshold is used to determine contiguous blocks of frames which are estimated to be sounding the same set of notes. The note prediction algorithm is performed on a subset of the frames in each block to sample the hypothesized notes. The number of notes for each frame in the block is taken to be the average number of notes from the sampled frames. Since the notes for each frame are estimated in decreasing prominence order, the notes for the block are estimated from the sampled frames by weighting the notes in decreasing value based on the order of the notes in the frame.

Each block then has one set of notes associated with it and all of the frames within that block are reported as those same notes.

This approach has not yet been extended to predict individual note onset and termination times as in track 2.

Table 1. Algorithm Results

corpus	corpus N_{ref}	P	R
guitar	1,521,816	.595	0.683
piano	229,986	0.723	0.674
development	19,300	0.533	0.491
MIREX 2008	256,390	0.481	0.570

4 RESULTS

The system was submitted with a model created from sampled piano and guitar notes from Apple's Logic Studio product. Four sampled pianos and eight sampled guitars were combined to come up with models for each of the MIDI pitches from 24 to 97. Testing on independently collected piano and guitar MIDI file corpora showed better performance than the development set. Not surprisingly, the ultimate MIREX results showed performance similar to that of the development set. Unfortunately, the piano selections used in track 2 are not scored for track 1 for comparison purposes. The details of track 2 do show however that the piano selections give much better results for the submitted algorithms than the full set of selections for track 2. This implies that the piano selections may be inherently "easier" to predict than the other instrument selections.

Table 1 shows the result of this algorithm and model against the various training corpora and final MIREX results. Eight of the 14 remaining submissions scored better than this submission with statistical significance.

4.1 Performance

The system was implemented in C++ using the vector operations provided in the Accelerate Framework in Mac OS X. Estimation of a one minute audio file takes about 1.5 seconds of (single-threaded) processing time on a 2.8 GHz Quad-Core Intel Xeon processor. The MIREX results were performed on a 1.83 Ghz Intel Core 2 Duo Mac with a runtime of 99 which was more than 9 times faster than the next fastest algorithm with a run time of 955.

5 REFERENCES

[1] A.P. Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *Speech and Audio Processing, IEEE Transactions on*, 11(6):804–816, Nov. 2003.

[2] S. Mukhopadhyay and G.C. Ray. A new interpretation of nonlinear energy operator and its efficacy in spike detection. *Biomedical Engineering, IEEE Transactions on*, 45(2):180–187, Feb 1998.

[3] Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP J. Appl. Signal Process.*, 2007(1):154–154, 2007.