PITCH CONTOUR TRACKING IN MUSIC USING HARMONIC LOCKED LOOPS

Rachel M. Bittner[1]*, Avery Wang*[2]*, Juan P. Bello*[1]

[1] Music and Audio Research Lab, New York University, [2] Shazam Inc.

ABSTRACT

We present a novel time-domain pitch contour tracking algorithm based on Harmonic Locked Loops, which differs from existing method in terms of its approach, resolution, timbre information and speed. In addition to estimating pitch contours, the proposed method computes the amplitude of each harmonic over time, expanding the potential set of features that can be used for higher level tasks such as melody extraction. The method is tested against ground truth melody pitch annotations from publicly available datasets, and we show that contour recall is improved compared with a state of the art approach.

Index Terms— pitch, melody, contour, music

1. INTRODUCTION

A pitch *contour* is a continuous trajectory of fundamental frequency values over time, whose length may vary from a single note in the shortest case to a short phrase in the longest. A number of MIR algorithms use pitch contours as a mid-level representation for other tasks, including melody extraction [1, 2, 3, 4, 5, 6] and multiple-f_0 estimation [7, 8, 9]. In melody extraction, for example, several popular methods first estimate a set of candidate contours and classify which belong to the melody. Thus, the objective of *contour tracking* is to estimate a set of contours from polyphonic music such that the contours belonging to the target concept (e.g. melody) are included. The most important goal of a contour tracking algorithm is to have high target contour recall, as incorrect contour estimates can be easily detected.

Existing contour tracking methods have been proposed based on sinusoidal modeling [10], phase vocoding [4], source-filter modeling [2] and time-frequency peak-streaming [11, 1]. These methods rely on a variety of input representations including multi-resolution transforms [4, 5], but have been found to be comparable to standard transforms for contour extraction [12], and are subject to the usual time-frequency resolution trade-off. Timbre information, which can be used as additional information to help determine whether a contour belongs to the melody, is implicitly encoded in some methods [2, 10], but most do not supply this information.

While many contour tracking methods have achieved good performance on certain datasets, these results do not generalize to expanded datasets, such as MedleyDB [13]. In [3], we performed an evaluation of the contour tracking algorithm proposed in [12] on MedleyDB. We found that only 66% ($\pm 22\%$) of ground truth melody contour frames were present in the set of contour candidates, imposing an upper bound on melody extraction performance. Recently the authors of [6] showed that replacing the contours used in several existing algorithms improved melody extraction performance on both MedleyDB and Orchset [14], highlighting the need for renewed exploration of contour tracking approaches.

In [15], a time-domain algorithm is proposed called a *Harmonic Locked Loop* (HLL), which is designed to separate continuous non-stationary harmonic signals in the presence of heavy interference, e.g. removing a singing voice from a symphony. An HLL-based contour tracking method has not yet been explored, and the HLL's robustness , computational efficiency, and timbre information makes it an interesting algorithm to employ for this task. Being a time-domain approach, the estimated contour frequencies are not subject to a particular grid, and the time resolution of the contours is at the audio sample-rate. Note that similar time-domain methods have been used for pitch tracking [16, 17], but are for monophonic, not polyphonic audio. Furthermore, an HLL estimates the amplitude of each harmonic over time, providing rich timbre information. The original HLL algorithm was originally designed for source separation, performing contour tracking as an intermediate step; the focus was on the quality of the source separation, and the algorithm was developed using very short test signals that required an accurate initial time-frequency seed and had no stopping criterion. The parameters of the original HLL were manually tuned to the test signals, and were not tested on or optimized for a larger corpus.

In this paper, we explore an HLL-based contour tracking method, which has very different intrinsic properties from existing contour tracking methods, and the potential to advance the state of the art. We modify the HLL algorithm to impose a stopping criterion, adapt several intermediate steps to better suit our task, include an automatic seed detection method to use as initial conditions to the tracker, and perform an extensive sweep over the algorithm's parameters on a corpus of varied musical data. Finally, we verify the effectiveness of the method by comparing it with a state of the art approach and

provide a run-time analysis.

2. BACKGROUND

2.1. Frequency Locked Loops

An HLL is an extension of a *Frequency Locked Loop* (FLL) [18]. Let $y[n]$ be a sinusoidal audio signal modeled as

$$y[n] = a[n] \exp\left(\frac{j2\pi}{f_s} \sum_{k=0}^{n} f[k] + \phi_0\right) + w[n] \tag{1}$$

where $f[n]$ and $a[n]$ are the instantaneous frequency and amplitude at time n, ϕ_0 is the initial phase, f_s is the sample rate, and $w[n]$ is an interfering signal. An FLL computes an estimate $\hat{f}[n]$ of an instantaneous frequency $f[n]$ through an iterative procedure. A single iteration of the frequency locked loop first consists of updating instantaneous frequency and phase estimates given the error ϵ from the previous sample and the tracking gain G:

$$\hat{f}[n] = \hat{f}[n-1] + G \cdot \epsilon[n-1] \tag{2}$$

$$\hat{\phi}[n] = \hat{\phi}[n-1] + \frac{2\pi}{f_s} \hat{f}[n] \tag{3}$$

The audio signal is demodulated to baseband by a complex sinusoid at the estimated frequency, shifting the spectrum of the signal by $-\hat{f}[n]$:

$$d[n] = y[n] \cdot \exp\left(-j \cdot \hat{\phi}[n]\right) \tag{4}$$

The demodulated signal d is low-pass filtered, resulting in a complex sinusoid u whose frequency is approximately equal to the error of the estimate $f[n] - \hat{f}[n]$:

$$u[n] = \frac{1}{\alpha_0} \left(\sum_i \beta_i d[n-i] - \sum_j \alpha_j u[n-j] \right) \tag{5}$$

where β_i and α_j are the lowpass filter coefficients[1]. If the estimate $\hat{f}[n]$ is close to $f[n]$, u will have instantaneous frequency that is close to zero. The frequency tracking error ϵ is computed as the change in phase of u, closing the loop:

$$\epsilon[n] = \frac{f_s}{2\pi} \cdot \angle(u[n] \cdot u[n-1]^*) \tag{6}$$

We may estimate the amplitude $\hat{a}$ of the tracked sinusoid as:

$$\hat{a}[n] = |u[n]| \tag{7}$$

2.2. Extending FLLs to HLLs

An FLL is able to accurately track frequencies for signals in which there is little interference in the frequency region within the cutoff frequency f_c of f. However in polyphonic music, there are often signals present in this frequency region caused by transients, harmonics from other pitched sounds, etc. An HLL addresses this by extending the FLL algorithm to incorporate information drawn from the first H harmonics as well; it runs a series of FLL trackers in parallel for each harmonic, and jointly computes the tracking update $\epsilon^\dagger$ as a weighted average of the tracking error produced by each harmonic's tracker where $\hat{v}[n,h]$ is the local variance of the tracker for harmonic h:

$$\epsilon^\dagger[n] = \sum_{h=0}^{H-1} (\hat{v}[n,h])^{-1} \epsilon[n,h] \Big/ \sum_{h=0}^{H-1} (\hat{v}[n,h])^{-1} \tag{8}$$

$$\hat{v}[n,h] = g_v \cdot \hat{v}[n-1,h] + (1-g_v)(\epsilon[n,h])^2 \tag{9}$$

for a fixed variance gain g_v. Since the weight for a tracker is inversely proportional to its local variance in tracking error, a tracker that is interrupted by noise is weighted less heavily in the final update than stable trackers. The tracking error for harmonic h is a scaled version of Eq. 6:

$$\epsilon[n,h] = \frac{f_s}{2\pi(h+1)} \angle(u[n,h]u[n-1,h]^*) \tag{10}$$

where $u[n,h]$ is computed as in Eq. 5, but using the value of d for harmonic h. The frequency update from Eq. 2 becomes:

$$\hat{f}[n] = \hat{f}[n-1] + G \cdot \epsilon^\dagger[n-1] \tag{11}$$

3. METHOD

In this section, we present a high-level overview of the contour tracking method, the modifications made to the HLL algorithm, and the automatic seed detection method. We then perform a sweep over the modified HLL parameters and provide a run-time analysis. The metrics are computed using `mir_eval` [19], and our code is made available online [2].

3.1. Contour Tracking

Algorithm 1 Track Contours

procedure TRACKCONTOURS(y, f_s)
  $N \leftarrow$ `len`(y)
  $y_r \leftarrow$ `Reverse`(y)
  `seeds` $\leftarrow$ `GetSeeds`(y, f_s)
  for n_0, f_{init} **in** `seeds`
    $c_+ \leftarrow$ HLL($y, f_s, n_0, f_{\text{init}}$)
    $c_- \leftarrow$ HLL($y_r, f_s, N - n_0, f_{\text{init}}$)
    `contour` $\leftarrow$ `Join`(`Reverse`(c_-), c_+)
    yield `contour`

The proposed contour tracking method consists of first estimating a series of time-frequency seeds, and performing tracking on each seed as shown in Algorithm 1. Seeds can be located anywhere in the contour, so we run a modified HLL both forward and backward in time to track the entire contour. The backward tracking is performed by simply running the

[1] We found that the HLL is quite sensitive to the type of filter, and it is imperative that the filter has a fast roll-off.

[2] https://github.com/rabitt/icassp-2017-hll

HLL on the reversed audio signal. The output of the algorithm is a list of contours, where each contour is a sequence of time-frequency-amplitude tuples $(t, f, a_0, a_1, \ldots a_{H-1})$, where a_i denotes the amplitude of harmonic i, giving us detailed timbre information. An HLL gives frequency estimates at the sample level, but tasks using contours rarely need this level of granularity, so the method decimates the output contours to a lower sample rate of $\frac{44100}{256} \approx 172$ Hz.

3.2. HLL Modifications

We first introduce a stopping condition that is triggered if either the tracking error $\epsilon^\dagger$ rises above a given threshold ϵ_{max} or the average amplitude $\tilde{a}$ falls below a minimum threshold a_{min}, both of which indicate that the tracking has become unstable. We compute the average amplitude similarly to $\epsilon^\dagger$:

$$\tilde{a}[n] = \sum_{h=0}^{H-1} (\hat{v}[n,h])^{-1} \hat{a}[n,h] \Big/ \sum_{h=0}^{H-1} (\hat{v}[n,h])^{-1} \tag{12}$$

When the tracker is first seeded these stopping conditions may be triggered while the tracker is stabilizing, so we additionally enforce a minimum contour length n_{min}.

Little attention is given to the tracking gain G in the original work, and the experiments leave it constant. However, since the tracking gain controls the amount the frequency can change between samples, we use $G = g \cdot \frac{f[n-1]}{440.0}$ (where g is a constant) such that the tracking gain scales with frequency, centered around a fixed frequency we choose to be 440 Hz. Finally, we use a 4th order Butterworth filter with a cutoff of $f_c = 30$ Hz in Eq. 5 because of its steep roll-off and minimal computational cost. The complete modified algorithm is outlined in Algorithm 2.

3.3. Seed Detection

We define a "seed" as a time/frequency coordinate that is given as an initial condition to the HLL. An ideal seed lies within f_c=30 Hz of the true frequency at any point in time where the contour is active. We design a liberal seed selection algorithm, intended to overestimate the number of seeds to maximize recall. Of course, a solution to maximize recall is to exhaustively select every time-frequency pair along some grid, but as we will discuss in Section 3.5, the computation time of the algorithm is linear in the number of seeds, thus we prune the selection to the most relevant candidates. The audio is first passed through a simple Harmonic-Percussive source separation algorithm [20] to reduce transients. A constant-Q representation of the resulting harmonic audio signal is computed and the amplitudes are normalized to be between 0 and 1. The amplitudes for each frequency band are passed through an averaging filter, and the seeds are chosen from the peaks across time of the resulting signals.

Algorithm 2 Modified Harmonic Locked Loop

procedure HLL($y, f_s, n_0, f_{init}, H, g, n_{\text{min}}, \epsilon_{\text{max}}, a_{\text{min}}$)
- $\beta, \alpha \leftarrow$ Butterworth($f_c = 30$ Hz, order $= 4$)
- $n = n_0$ ▷ starting index
- $C = 1$ ▷ counter
- $\hat{f}[n] = f_{init}$ ▷ frequency estimate
- $\hat{\phi}[n] = 0.0$ ▷ phase estimate
- $\epsilon^\dagger[n] = 0.0$ ▷ tracking update
- $\tilde{a} = 1.0$ ▷ current average amplitude
- **for** $h = 0$ **to** $H - 1$
  - $\epsilon[n,h] = 0.0$ ▷ tracking error
  - $\hat{v}[n,h] = 1.0$ ▷ tracking variance
  - $\hat{a}[n,h] = 0.0$ ▷ amplitude estimate
  - $d[n,h] = y[n_0], \quad u[n,h] = 0.0$
- **while** $C < n_{\text{min}}$ **or** ($\tilde{a} \geq a_{\text{min}}$ **and** $\epsilon^\dagger[n] \leq \epsilon_{\text{max}}$)
  - $C = C + 1, \quad n = n + 1$
  - $\hat{f}[n] \leftarrow$ Eq (11), $\hat{\phi}[n] \leftarrow$ Eq (3)
  - **for** $h = 0$ **to** $H - 1$
    - **if** h **is** 0 **then**
      - $d[n,h] \leftarrow$ Eq (4)
    - **else**
      - $d[n,h] \leftarrow d[n,h-1] \cdot \exp(-j \cdot \hat{\phi}[n])$
    - $u[n,h] \leftarrow$ Eq (5), $\epsilon[n,h] \leftarrow$ Eq (10)
    - $\hat{v}[n,h] \leftarrow$ Eq (9), $\hat{a}[n,h] \leftarrow$ Eq (7)
  - $\tilde{a} \leftarrow$ Eq (12), $\epsilon^\dagger[n] \leftarrow$ Eq (8)
- **return** $\hat{f}, \hat{a}$

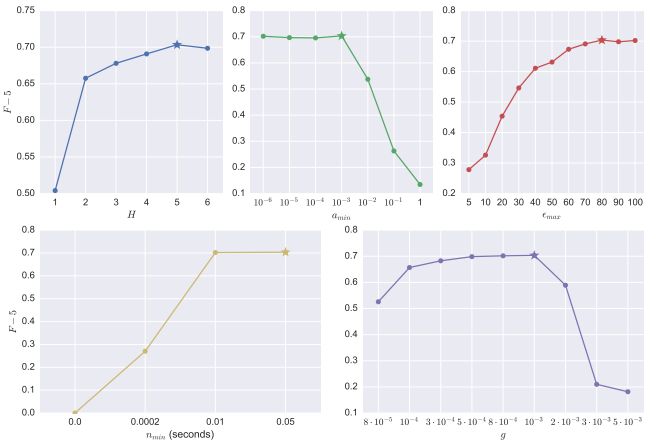

Fig. 1. Maximum achieved F_β score for each HLL tracking parameter across all other combinations of parameter settings. The best configuration found was $H = 5$, $a_{\text{min}} = 0.001$, $\epsilon_{\text{max}} = 100$, $n_{\text{min}} = 0.05$ s, and $g = 0.001$.

3.4. Parameter Settings

While [15] provides an extensive theoretical background on the HLL method and some qualitative examples, relatively little objective testing on real data was performed. We perform a 2000 iteration randomized sweep of the modified HLL parameters H, a_{min}, ϵ_{max}, n_{min}, and g jointly on 7 randomly selected tracks from the MedleyDB dataset [13] using ground truth seeds as input to the HLL. The seeds were computed from the dataset's annotated pitch contours [3], and were chosen as

[3] using the Melody type-3 annotations, see [13] for details

the time frequency point at the center of each contour. Pitch annotations that jumped more than 25 cents in frequency between frames were split into multiple contours. We evaluate the quality of the output contours by computing the precision (percentage of predicted frames that were within a semitone of the ground truth) and recall (percentage of ground truth frames that had an estimate within a semitone of the ground truth) as described in [21]. The optimal parameters were chosen as those with the highest average F_β score. For our purposes, recall is more important than precision, as the output of a contour extraction algorithm can be cleaned up by a selection method [3, 6], and we use $\beta = 5$. The results for the highest achieved average F_β for each parameter value are reported in Figure 1, and the optimal parameter is marked with a diamond. We see that performance improves with the number of harmonics, as we would hope, and drops off slightly at $H = 6$. The minimum contour length is unsurprisingly optimal at the longest allowed value, and we do not allow it to be longer than a quarter second to prevent artificially long contours.

3.5. Algorithmic Complexity and Runtime

The HLL method is an $O(H \cdot C)$ algorithm, where H is the number of harmonics and C is the number of samples in an estimated contour. C is variable depending on the type of data, but is roughly 1 second per contour across the parameter tuning set. We use $H = 5$ in our experiments, but runtime can be reduced greatly by lowering H for only a slight drop in accuracy, shown in Figure 1, top left. The `TrackContours` method is $O(H \cdot C \cdot M)$, where M is the number of seeds, and the average track in MedleyDB has ≈ 230 ground truth melody 3 seeds.

We test the HLL's runtime using a random subset of estimated seeds. The average runtimes per seed were 23.8 ± 9.0, 19.5 ± 10.8, and 19.0 ± 10.4 milliseconds for 10, 100, and 1000 seeds, respectively[4]. An interesting trait of the runtime of `TrackContours` is that it is not directly a function of the length of a recording, but rather the number of seeds. The number of seeds is correlated to but not dependent on the length of the audio - a 30 second symphony recording will likely have more seeds than a 2 minute solo voice recording.

4. EXPERIMENTS

In the following experiments, we run `TrackContours` on the remaining 61 tracks in MedleyDB with melody annotations, and evaluate the output using the recall and precision metrics defined in [21]. Additionally we run another contour tracking algorithm (which we will refer to as SAL) described in [12]. There is a direct relationship between contour recall (against ground truth *melody* contours) and the upper bound on overall accuracy for a melody extraction system. (For details on the referenced melody extraction metrics see [1]). Given a set of contours with contour recall r, a system that correctly estimates voicing and selects the best possible contours out of the set of estimates will achieve an overall accuracy of r, and a raw pitch accuracy of r. Similarly, if the above set of contours had chroma recall r_c, an ideal system would achieve a raw chroma accuracy of r_c. There is no direct relationship between contour precision and melody extraction metrics - better precision simply means that the contour classifier has fewer estimates to select from.

For an upper bound on the HLL's performance, we first compute contours using ground truth seeds. The resulting contours had a recall of 0.69 (± 0.11) and a precision of 0.33 (± 0.11), compared with SAL which had a recall of 0.59 (± 0.19) and precision of 0.31 (± 0.15). We repeated the above experiment using estimated seeds as described in Section 3.3, which achieved an accuracy of 0.63 (± 0.19) and precision of 0.04 (± 0.04). We see that this still outperforms SAL, but is a bit below the upper bound due to missed seeds. The overestimated number of seeds also leads to quite low precision in this setting, but as discussed in Sec. 1, low precision is not a big issue because the majority of the incorrect contours estimated by an HLL can be easily identified by a simple classifier [3].

We additionally examine the recall when all frequencies are wrapped to a single octave (referred to as chroma), forgiving octave mistakes. The upper bound for HLL's chroma recall was 0.70 (± 0.12) (versus 0.69 recall), indicating only 1% of the mistakes were octave errors. With estimated seeds, chroma recall was 0.81 (± 0.14), thus nearly a fifth of the mistakes made by the tracker were octave mistakes, likely due to seeds that were estimated at octaves above/below the true contour. SAL falls in between, with a chroma recall of 0.67 (± 0.16).

Overall, we find that there is still a glass ceiling on the performance of algorithms using HLL contours as input because the algorithm does not achieve perfect recall. However, the aforementioned glass ceiling is higher than before, as the HLL performs better than SAL in terms of contour recall.

5. CONCLUSIONS AND FUTURE WORK

We presented a novel time domain contour tracking algorithm based on a modified harmonic locked loop, showed that it outperforms a state of the art method in terms of melody contour recall, and provided a runtime analysis.

The increase in melody recall addresses the bottleneck in current melody extraction algorithms, and could be used to advance the state of the art. Finally, we plan to utilize these contours as inputs for melody extraction, and explore the use of the time varying harmonic amplitudes as features for melody extraction and instrument identification, or clustering/streaming tasks.

[4] Runtimes were measured on a 2014 MacBook Pro with a 2.6 GHz Intel Core i5 processor, 8GB DDR3 RAM, and a 250GB SSD.

6. REFERENCES

[1] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, Aug. 2012.

[2] J.-L. Durrieu, B. David, and G. Richard, "A musically motivated mid-level representation for pitch estimation and musical audio source separation," *IEEE Journal on Selected Topics on Signal Processing*, vol. 5, no. 6, pp. 1180–1191, Oct. 2011.

[3] Rachel M Bittner, Justin Salamon, Slim Essid, and Juan P Bello, "Melody extraction by contour classification," in *International Society of Music Information Retrieval (ISMIR)*, October 2015.

[4] K. Dressler, "Sinusoidal extraction using an efficient implementation of a multi-resolution FFT," in *Proc. 9th Int. Conf. on Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sep. 2006, pp. 247–252.

[5] P. Cancela, M. Rocamora, and E. López, "An efficient multi-resolution spectral transform for music analysis," in *10th Int. Soc. for Music Info. Retrieval Conf.*, Kobe, Japan, Oct. 2009, pp. 309–314.

[6] Juan Jos Bosch Vicente, Rachel M. Bittner, Justin Salamon, and Emilia Gmez Gutirrez, "A comparison of melody extraction methods based on source-filter modelling," in *International Society of Music Information Retrieval (ISMIR)*, August 2016.

[7] Anssi P Klapuri, "Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes," in *Proceedings of the International Society of Music Information Retrieval*, 2006.

[8] Zhiyao Duan, Bryan Pardo, and Changshui Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 8, pp. 2121–2133, 2010.

[9] Chunghsin Yeh, Axel Röbel, and Xavier Rodet, "Multiple fundamental frequency estimation of polyphonic music signals," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. IEEE, 2005, vol. 3, pp. iii–225.

[10] Xavier Serra and Julius Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[11] K. Dressler, "Pitch estimation by the pair-wise evaluation of spectral peaks," in *AES 42nd Int. Conf.*, Ilmenau, Germany, Jul. 2011, pp. 278–290.

[12] J. Salamon, E. Gómez, and J. Bonada, "Sinusoid extraction and salience function design for predominant melody estimation," in *Proc. 14th Int. Conf. on Digital Audio Effects (DAFx-11)*, Paris, France, Sep. 2011, pp. 73–80.

[13] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *International Society of Music Information Retrieval (ISMIR)*, October 2014.

[14] Juan J Bosch, Ricard Marxer, and Emilia Gómez, "Evaluation and combination of pitch estimation methods for melody extraction in symphonic classical music," *Journal of New Music Research*, pp. 1–17, 2016.

[15] Avery Wang, *Instantaneous and frequency-warped signal processing techniques for auditory source separation*, Ph.D. thesis, Stanford University, 1994.

[16] Johannes Böhler and Udo Zölzer, "Monophonic pitch detection by evaluation of individually parameterized phase locked loops," in *19th International Conference on Digital Audio Effects (DAFX16)*, 2016.

[17] Hideki Kawahara, Yannis Agiomyrgiannakis, and Heiga Zen, "Using instantaneous frequency and aperiodicity detection to estimate f0 for high-quality speech synthesis," *arXiv preprint arXiv:1605.07809*, 2016.

[18] Kenneth K Clarke and Donald T Hess, "Frequency locked loop FM demodulator," *Communication Technology, IEEE Transactions on*, vol. 15, no. 4, pp. 518–524, 1967.

[19] C. Raffel, B. McFee, E. Humphrey, J. Salamon, O. Nieto, D. P. W. Ellis, and D. Liang, "mir eval: A Transparent Implementation of Common MIR Metrics," in *International Society for Music Information Retrieval Conference*, 2014.

[20] D. Fitzgerald, "Harmonic/percussive separation using median filtering," in *13th International Conference on Digital Audio Effects (DAFX10)*, 2010.

[21] Mert Bay, Andreas F Ehmann, and J Stephen Downie, "Evaluation of multiple-f0 estimation and tracking systems.," in *ISMIR*, 2009, pp. 315–320.