MELODY EXTRACTION BY CONTOUR CLASSIFICATION

Rachel M Bittner, Justin Salamon, Slim Essid, Juan P Bello

MELODY EXTRACTION BY CONTOUR CLASSIFICATION

Rachel M. Bittner[1], Justin Salamon[1,2], Slim Essid[3], Juan P. Bello[1]
[1]Music and Audio Research Lab, New York University
[2]Center for Urban Science and Progress, New York University
[3]Télécom Paris-Tech
{rachel.bittner,justin.salamon,jpbello}@nyu.edu {slim.essid}@telecom-paristech.fr

ABSTRACT

Due to the scarcity of labeled data, most melody extraction algorithms do not rely on fully data-driven processing blocks but rather on careful engineering. For example, the Melodia melody extraction algorithm employs a pitch contour selection stage that relies on a number of heuristics for selecting the melodic output. In this paper we explore the use of a discriminative model to perform purely data-driven melodic contour selection. Specifically, a discriminative binary classifier is trained to distinguish melodic from non-melodic contours. This classifier is then used to predict likelihoods for a track's extracted contours, and these scores are decoded to generate a single melody output. The results are compared with the Melodia algorithm and with a generative model used in a previous study. We show that the discriminative model outperforms the generative model in terms of contour classification accuracy, and the melody output from our proposed system performs comparatively to Melodia. The results are complemented with error analysis and avenues for future improvements.

1. INTRODUCTION

Melody extraction has a variety of applications in music retrieval, classification, transcription and analysis [15]. A precise definition of melody that takes into account all possible scenarios has proven elusive for the MIR community. In this paper we consider two different definitions of melody [1]: The f_0 curve of the predominant melodic line drawn from a single source (melody type 1), and the f_0 curve of the predominant melodic line drawn from multiple sources (melody type 2).

Some approaches to melody extraction are source separation-based [4, 18], first isolating the melodic source from the background and then tracking the pitch of the resulting signal. The most common approaches are based on the notion of salience [3, 7, 13, 14], and are variants of the following steps (1) audio pre-processing, (2) salience function computation, (3) f_0 tracking, and (4) voicing decisions. Steps (3) and (4) for these methods are each based on a series of carefully chosen heuristic steps, and are limited to the data they were designed for. A recent trend in Music Information Retrieval research is to combine domain knowledge with data driven methods [8], using domain informed feature representations as input to data-driven models. To the best of our knowledge, only one melody extraction approach [5] has been proposed to date using a fully data driven method. However, the features employed were poor for the task (magnitude Fourier coefficients), and used only limited temporal modeling via HMM smoothing. Additionally, at the time, only a small amount of data was available. The recent availability of annotated melody data allows for new exploration into data driven methods for melody extraction.

In this paper, we present a system for melody extraction which replaces the common series of heuristic steps with a data-driven approach. We propose a method for scoring extracted contours (short, continuous pitch sequences) using a discriminative classifier, and a Viterbi-based method for decoding the output melody. We show that our method performs competitavely with Melodia [14]. The implementation of the proposed method and the code used for each experiment is available on Github [1] . The remainder of this paper is organized as follows: in Section 2 we give an overview of Melodia; Section 3 describes our proposed method for melody extraction; in Section 4 we present experiments evaluating the effectiveness of our method, including a comparison with Melodia, and in Section 5 we discuss the conclusions and avenues for future work.

2. MELODIA

Melodia [14], a salience-based melody extraction algorithm, has proved to perform particularly well. The algorithm is comprised of four processing blocks: sinusoid extraction, salience function computation, contour creation and characterization, and finally melody selection. In the first block, spectral peaks are detected, and precise frequencies of each component are estimated using their instantaneous frequency. In the second stage a harmonic summation-based salience function is computed. In the third block, the peaks of the salience function are tracked into continuous pitch contours using auditory streaming

[1] www.github.com/rabitt/contour_classification

cues. Additionally, a number of features are computed for each contour:

- Duration (seconds)
- Pitch mean and standard deviation (in cents)
- Salience mean, standard deviation, and sum
- Vibrato presence (binary), rate (Hz), extent (cents), coverage (fraction of contour with vibrato)

The melody f_0 trajectory is obtained in the fourth block by filtering out non-melodic contours based on their features in combination with an iterative estimation of the global melodic trend. This step exploits the contour feature distributions to perform the filtering, but does so in a heuristic fashion. For further details the reader is referred to [14].

Recently, Melodia was evaluated on the MedleyDB [1] dataset which contains considerably more variety in musical style than previous datasets. The results were shown to be substantially lower than for the existing datasets. In particular, it was reported that Melodia's performance on music with vocal melodies was better than on music with instrumental melodies. This indicates that the heuristic steps at the contour selection stage may be well tuned for singing voice, but less so for instrumental melodies. Since these heuristics are hard coded, the algorithm cannot be adjusted for different kinds of data or different concepts of melody. We will show that these steps can be replaced by a data driven approach.

In [16], Salamon, Peeters, and Röbel proposed to replace the heuristic melodic selection block of Melodia with a generative classification model. The contour features were used to estimate two multivariate Gaussian distributions, one for melodic and another for non-melodic contours. These distributions were used to define the "melodiness" score, computed as the likelihood ratio of the two distributions.

The final f_0 sequence was obtained by taking the f_0 of the contour with the highest "melodiness" at each frame. The authors showed that the generative model could produce similar (albeit not equally good) results in terms of pitch accuracy, but the model lacked a voicing detection step. This was addressed by combining the model with the voicing detection filter of the original Melodia algorithm.

Finally, in [17] the authors combined Melodia's contour features with additional features to train a discriminative model for classifying different musical genres. Their experiments showed that the contour features carry discriminative melodic information. This outcome, together with that of [16] and the release of MedleyDB, gives compelling motivation for the exploration of discriminative models using pitch contour features for solving the problem of melodic contours selection.

3. METHOD

The proposed system uses the pitch contours and contour features generated by Melodia [2]. The method consists of a contour labeling stage, a training stage where a classifier is fit to discriminate melody from non-melody contours, and a decoding stage which generates a final f_0 sequence. Melody output is computed using a trained classifier as shown in Figure 1.

3.1 Contour Labeling

To generate contours for a musical audio signal, we use the first three processing blocks of the Melodia algorithm directly (see [14] for details). Each contour is represented by a sequence of tuples (time, frequency, salience). As described in Section 2, the third block also computes a set of descriptive features for each contour, which we use to train the model in Section 3.2.

During training, extracted contours are assigned binary labels: 1 if the contour should be considered as a part of the melody and 0 otherwise. The labels are chosen by comparing the amount of overlap between each contour and the ground truth annotation. Given an annotation $a(t)$ with $0 \leq t \leq T$, a contour $c(t)$ spanning the time interval $t_1 \leq t \leq t_2$ is compared with $a(t)$ over the time range $t_1 \leq t \leq t_2$. The amount of overlap between these two sequences is computed using "Overall Accuracy" [15], defined as:

$$\text{Acc}_{\text{ov}} = \frac{1}{L} \sum_{i=0}^{L-1} v_i \mathcal{T}\left[|\hat{\varphi}_i - \varphi_i|\right] + (1 - v_i)(1 - \hat{v}_i) \quad (1)$$

where L is the number of reference/estimate examples, v_i and $\hat{v}_i$ are the (binary) voicings of the reference and estimate respectively, φ_i and $\hat{\varphi}_i$ are the f_0 values in cents of the reference and estimate respectively, and $\mathcal{T}$ is a threshold function equal to 1 if the argument is less than 0.5, and 0 otherwise. Given a minimum overlap threshold α, if $\text{Acc}_{ov} > \alpha$ the contour is labeled as melody. Note that if $\alpha = 1$, because of the strict inequality, all contours would be labeled as non-melody. Despite containing extraneous information, a contour with a small degree of overlap still contains part of the melody. Labeling it as non-melody removes any possibility of the melody-portion of the contour ending up in the final extracted melody (i.e., lower recall). On the other hand, labeling it as melody potentially results in having non-melody information included in the melody (i.e., lower precision). Thus, there is an inherent trade-off between melody precision and recall based on the value of the overlap threshold α.

3.2 Contour Classification

We normalize the features per track to remove variance caused by track-level differences. The salience features are each divided by the maximum salience value in the track to remove differences based on overall track salience. The duration feature is normalized so that across the track the minimum value is 0 and the maximum value is 1. The feature "total salience" is additionally re-scaled to reflect the normalized duration.

These features and the computed labels are used to train a random forest classifier [2]. We use the random forest implementation in `scikit-learn` [11] with 100 trees and

[2] These are taken from intermediate steps in the Vamp plugin's implementation

Figure 1. Block diagram of the proposed system (left to right): pitch contours are extracted from an audio signal, a classifier is used to score the contours and remove those below a threshold, the final f_0 sequence is obtained using Viterbi decoding.

choose the maximum depth parameter by cross validating over the training set. In our experiments, the classifier was trained with roughly 11,000 examples for melody 1 and roughly 15,000 for melody 2. Because our class distributions tend to be biased towards non-melody examples, the classifier is trained with class weights inverse to the class distributions. Once the classifier is trained, we use it to predict the probability that a given contour is melody. In the case of a random forest, the melody likelihood is computed as the fraction of trees that classify a given example as melody.

3.3 Melody Decoding

We create an output melody by first removing contours whose likelihood falls below a threshold β and then decoding to generate a final melody. The thresholding step is necessary because there may be regions of time where only non-melody contours are present. Since decoding only chooses the best path through available contours, having regions with contours which are all non-melody would result in false positives. Aside from the contour extraction, the choice of this threshold is the single most important step for determining the voicing of the output melody.

This raises the question: what is the best way to determine the likelihood threshold β? A natural choice is $\beta = 0.5$, as this is the threshold that has been optimized by the machine for maximum classification accuracy. While this threshold gives us nearly perfect precision for the melody class, the recall is extremely low. We instead choose the threshold that yields the highest class-weighted F1 score on a validation set. The chosen value of β in this manner is consistently much lower than 0.5 (typically $\beta \approx 0.1$), resulting in higher recall at the cost of lower precision. It is interesting to note that for our end goal – selecting a single melody sequence – we do not necessarily need perfect precision because false positives can be removed during decoding.

After this filtering step, contours that do not overlap with any other contour are immediately assigned to the melody. The remaining contours have partial overlap with at least one other contour, requiring the melody line to be chosen from within the overlapping segments. Thus, we divide these remaining contours into groups: contours $\{C_1[t], \ldots, C_n[t]\}$ each spanning some time interval are assigned to the same group if the union of their intervals forms a contiguous interval.

The path over time through each group of contours is computed using Viterbi decoding [6]. Given a group of n contours, our state space is the set of contour numbers $\{1, 2, \ldots, n\}$. We create a matrix Y of emission probabilities using each contour's likelihood score $[p_1, p_2, \ldots, p_n]$:

$$Y_{it} = \begin{cases} p_i & \text{if } C_i \text{ is active at time } t \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The transition matrix A, defined to encourage continuity in pitch space, is computed for each group as:

$$A_{ij} = \frac{\sum_{k \neq j}^{n} |\log_2(f_i) - \log_2(f_k)|}{(n-1)\sum_{k=1}^{n} |\log_2(f_i) - \log_2(f_k)|} \quad (3)$$

where f_i is the average frequency (in Hz) of contour i. This transition matrix, simply put, assigns a high transition probability between contours whose (log) frequencies are near one another, and a lower transition probability between contours which are far from one another in log frequency. The prior distribution is set to be uniform. Given the sequence of contour states $S[t]$ computed by Viterbi, for each time point t, the frequency $C_{S[t]}[t]$ is assigned to the melody.

4. EXPERIMENTS

For each of the following experiments we use the MedleyDB Dataset [1]. Of the 122 tracks in the dataset, we use the 108 that include melody annotations. We create train/test splits using an artist-conditional random partition (i.e., tracks from the same artist cannot be in both the train and test set). The complete training set is further split randomly into a training and validation. A given train, validate, and test split contains roughly 78%, 7%, and 15% respectively of the 108 tracks. We repeat each experiment with five different randomized splits to get a sense of the variance of the results when using different data. In Figures 2, 3, and 4, vertical lines indicate the standard deviation over the five splits. Recall that we consider two definitions of melody (Section 1). Consequently, when we report scores for melody type 1, the classifier was trained using the melody 1 annotations, and likewise for melody type 2. All evaluation metrics were computed using `mir_eval` [12].

4.1 Experiment 1: Generative vs. Discriminative Contour Classification

Before evaluating components of the proposed system, we first examine the recall of Melodia's contour extraction on this dataset. That is, given all extracted contours, what is the percentage of the reference melody that is covered by the contours (in terms of pitch overlap)? We tested this by selecting the "oracle" (i.e., best possible) f_0 curve from the

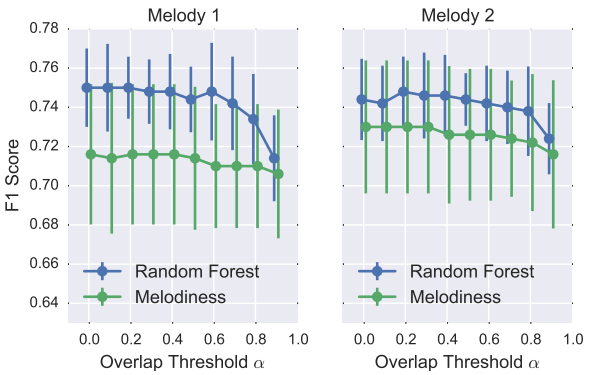

Figure 2. Maximum F1 score achieved per overlap threshold α by the generative and discriminative models.

contours. The oracle output yielded an average Raw Pitch Accuracy of 0.66 ($\sigma = 0.22$) for melody type 1, and 0.64 ($\sigma = 0.20$) for melody type 2. Thus, the best raw pitch accuracy we (or Melodia) could hope to achieve on this dataset is upper bounded by these numbers.

Acknowledging the existence of this glass ceiling, we compare the generative model for scoring contours with the proposed discriminative model. The features used for the discriminative model are those described in Section 2, while the generative model used only the continuous features (i.e., none of the vibrato features) [3] . The features for the multivariate Gaussian were transformed using the box-cox transformation as in [16], where the transformation's parameter λ was estimated from the training set and applied to the testing set.

To evaluate the effectiveness of these two methods, we compare the F1 scores achieved by selecting the optimal likelihood threshold β. Figure 2 shows the best achieved F1 scores on the validation set for the two models. We see that the random forest classifier obtains a better F1 score for all values of α. Interestingly, the F1 score achieved by the multivariate Gaussian is less affected by α than the Random Forest, which decreases as α increases. Note that neither classifier achieves an F1 score above 75%. This suggests that either the models are not complex enough or that the classes are not completely discriminable using this set of features. Since our feature set is relatively small, the latter is likely, and the performance of both of these models would likely benefit from a larger feature set. However, fitting a high dimensional multivariate Gaussian requires a large amount of data. Thus, another advantage of using a random forest classifier is that increasing the dimensionality of the feature space does not necessarily require more data.

One might argue that the difference in performance of the two methods could be due to the fact that the vibrato features are not used in the multivariate Gaussian model. However, a post-hoc analysis of the importance of the vibrato features within the random forest classifier (for melody 1 with $\alpha = 0.5$) showed that they were by a large margin the least important features in the set. In fact, the presence of vibrato contributed to discriminating only $\approx 0.03\%$ of the training samples. The most discriminative features for the random forest were the salience standard deviation, followed by pitch mean, followed by pitch standard deviation.

[3] We initially included the vibrato features for the generative model, but the results were extremely poor.

Melody Type	OA	RPA	RCA
1	1.6	2.5	2.5
2	2.4	4.2	2.1

Table 1. Percentage point difference between Viterbi decoding and taking a simple maximum.

Overall, we see that the random forest consistently outperforms the multivariate Gaussian, and has the additional benefit of scalability to a much larger feature set.

4.2 Experiment 2: Decoding Method

Our second experiment examines the effect of our Viterbi decoding strategy. First, we compare it with an approach based on the one used in [16], where the f_0 value at each point in time was chosen by taking a simple maximum over the "melodiness" score. For our comparison, we take the maximum over the likelihoods produced by the classifier after thresholding.

We found that Viterbi decoding consistently showed an improvement in the melody evaluation metrics on each track. For some particular tracks, Viterbi decoding improved the output by up to 10 percentage points. Table 1 shows the average percentage point increase per track by using Viterbi over the simple maximum. The metrics shown are the Overall Accuracy (OA), Raw Pitch Accuracy (RPA), and Raw Chroma Accuracy (RCA) [15]. We see a particularly good improvement for melody 2, where Viterbi decoding increases the average raw pitch accuracy by more than 4 percentage points.

Figure 3 shows each melody evaluation metrics across the different overlap thresholds α. The values plotted are averages over each of the 5 experiments, where the error bars indicate the standard deviation. Surprisingly, we see very little difference in any of the metrics for both melody types. We saw in Figure 2 that the F1 score decreased as α increased, which implies that unlike what we might expect, the final melody output is not strongly affected by the F1 score. Note, however, that the F1 score is computed on a different set of labels for each value of α. The resilience may be due to the fact that the labels that change as we sweep α are the "noisier" labels, and thus the hardest to classify, whereas the contours that are not affected by the value of α (i.e., very high overlap or no overlap with the annotation) are easier to classify. We conjecture that for each value of α the classifiers are probably performing equally poorly on the noisy contour examples and equally well on the clean examples.

All in all, the deviations in metrics are minor across values of α, and we conclude that the value of α does not have a strong impact on the final melody accuracy. The values

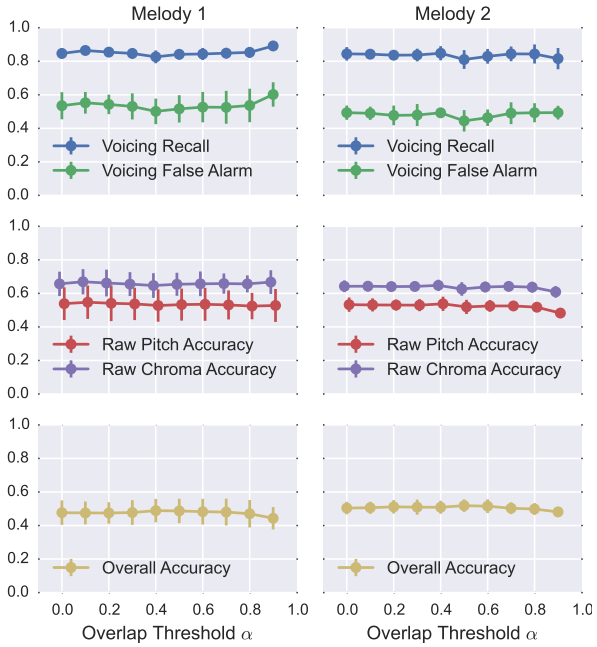

Figure 3. Melody metrics for each overlap threshold α and melody type.

of α that yield the highest scores on the validation set (by a small margin) were $\alpha = 0.5$ for melody 1 and $\alpha = 0.4$ for melody 2, and these values are used for our final system.

4.3 Experiment 3: Melodia vs. Proposed Method

As a final experiment, we compare the proposed method with Melodia. This experiment essentially evaluates the two different contour selection methods, since both methods begin with the same set of contours. Melodia's parameter ν controls the voicing decision, and is the parameter with the largest impact on the final output. The scores reported for Melodia in this experiment use the value of ν that achieved the best overall accuracy on the training set. The final scores are reported for the test set.

The results for each algorithm are shown in Figure 4. The proposed method performs quite competitively with Melodia. In particular, Melodia only outperforms our system in overall accuracy by 4 percentage points for melody 1 and 2 percentage points for melody 2. The primary metric where the algorithms differ is in the voicing recall and voicing false alarm rates. Our system has significantly better recall than Melodia (33 percentage points higher for melody 1, 9 for melody 2), but also a much higher false alarm rate (34 percentage points higher for melody 1, 14 for melody 2) - in other words, our system assigns contours to the melody much more often than Melodia does.

An interesting example to this point is shown in Figure 5. Both methods achieve the same overall accuracy of ≈ 0.50, but their output is quite different. Our output gets almost all of the voiced frames correct, but has spurious mistakes outside of the annotation as well. In contrast, Melodia has nearly perfect precision, but misses large segments. This example is characteristic of the difference between the two algorithms – our approach over-predicts melody, and Melodia under-predicts it. Notice that the proposed method produces spurious frequency values, caused by slight differences in contour start and end points within contour groups. These values could be removed in a future post processing stage.

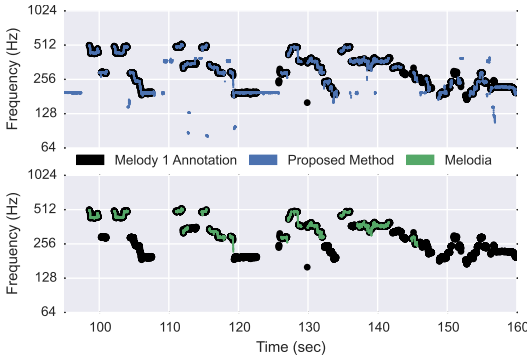

Figure 5. Segment of melody output for "Clara Berry and Wooldog: The Bad Guys".

This difference is especially significant for tracks containing instrumental melodies. Figure 6 shows a segment from a track with a flute melody. Our approach works quite well for this example, while Melodia misses most of the melodic line. In Figure 4 we also report the overall accuracy for the portions of the data containing vocal (OA-V) and instrumental melodies (OA-I). We see that for instrumental melodies, our method matches Melodia's performance for melody 1 and slightly outperforms Melodia for melody 2. Conversely, we see the opposite trend for vocals, with Melodia outperforming our method for both melody types. This trend can be largely attributed again to the differences in voicing statistics – vocal melodies in this dataset tend to have many more unvoiced frames than instrumental melodies, so our method's high false alarm rate hurts our performance for vocal tracks.

Despite the slight difference in metrics, the two algorithms perform similarly, with inversely related pitfalls. It is interesting to note that when the current approach completely fails, so does Melodia. This first and foremost occurs when output from the contour extraction stage is poor, which dooms both methods to failure.

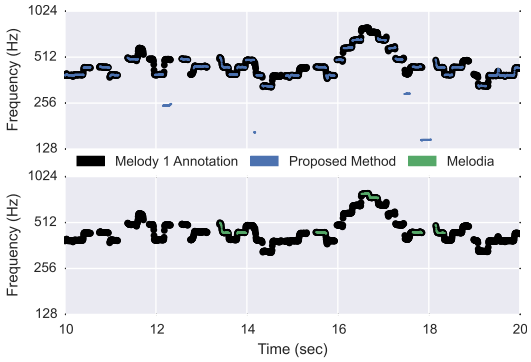

Figure 6. Segment of melody output for "Phoenix: Lark on the Strand/Drummond's Castle".

Overall, we see that the proposed method is quite good at correctly choosing melody examples, but the high false

Figure 4. Final melody output scores for the proposed method and Melodia. The metrics are abbreviated on the x-axis as: VR = Voicing Recall, VFA = Voicing False Alarm, RPA = Raw Pitch Accuracy, RCA = Raw Chroma Accuracy, OA = Overall Accuracy, OA-V = Overall Accuracy – vocal tracks, and OA-I = Overall Accuracy – instrumental tracks.

alarm rates hurt its overall scores. This speaks to the classifier's need for better discrimination between melody and non melody examples. To do this, we need more/better features, a more powerful classifier, or both. This ties back to Ellis and Poliner's observation in [5]: a large percentage of contours are very easy to distinguish, and the remaining contours are difficult for data driven and heuristic methods alike. This is likely due to the lack of longer time scale features describing the relationship between observations. We as humans are able to distinguish melody from non-melody in a song, but in ambiguous cases, we make our distinction based on what we heard earlier in the song [10].

As a final illustration, Figure 7 shows the output of both algorithms for melody 1 (top) and melody 2 (bottom) for a segment containing a flute and a trumpet. The melody is carried by the flute for most of the track, but in this segment is carried by the trumpet. For melody 1, both methods track the flute, matching the annotation, whereas for melody 2 both methods still track the flute whereas the trumpet line is annotated. Without long-term context giving the algorithm information about which lines have happened previously as background or melody, there is no way for either of these methods to choose the "correct" line.

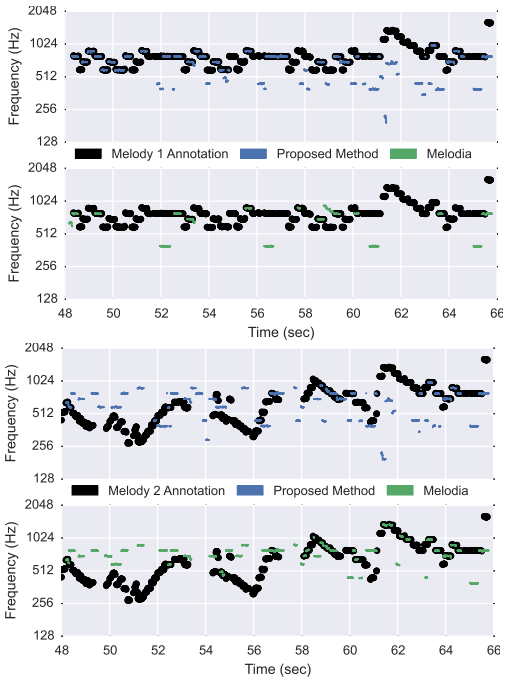

Figure 7. Outputs for melody 1 (top) and melody 2 (bottom) for a segment of "Music Delta: Latin Jazz".

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have shown that replacing Melodia's heuristic decisions with a series of fully data driven decisions can nearly match Melodia's overall performance, and there many open avenues for improving our results. In particular, we have shown that a discriminative model can outperform a generative model for labeling contours, and we have provided a detailed evaluation of how each step in the proposed approach influences the final results. Compared to Melodia, we noted that the proposed method has better melody recall, but a considerably worse voicing false alarm rate. To improve the discrimination ability of the classifier, future iterations of this method will first incorporate a wider set of features, including features that describe neighboring contours (octave duplicates, etc.), and features that describe a contour's relationship with the rest of the track on a longer time scale, potentially including timbre similarity. Additionally, since we are using a relatively small training set, we would like to explore augmenting our training data through sets of time and pitch deformations.

With a slight adjustment to the evaluation metrics, our method can be easily extended to be trained on and predict melody type 3 [1] annotations, which give all feasible melody candidates at each time point, and is the most inclusive melody definition for MedleyDB. A limitation of the current method is that it assigns a single likelihood to each contour. Since the extracted contours virtually never overlap completely with the annotation, it would be desirable to be able to assign time-varying scores to each contour. To do this, we plan to explore the use of Conditional Random Fields [9] for assigning scores to contours because of their ability to incorporate temporal information. Finally, to raise the glass ceiling on performance, future work will include revisiting the contour extraction stage.

6. REFERENCES

[1] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. MedleyDB: a Multitrack Dataset for Annotation-Intensive MIR Research. In *International Society for Music Information Retrieval Conference*, July 2014.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] K. Dressler. An Auditory Streaming Approach for Melody Extraction from Polyphonic Music. In *International Society for Music Information Retrieval Conference*, 2011.

[4] Jean-Louis Durrieu, Gaël Richard, Bertrand David, and Cédric Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(3):564–575, March 2010.

[5] D. P. W. Ellis and G. Poliner. Classification-Based Melody Transcription. *Machine Learning Journal*, 65(2-3):439–456, December 2006.

[6] G. D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[7] M. Goto. A Real-Time Music-Scene-Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-World Audio Signals. *Speech Communication*, 43(4):311–329, September 2004.

[8] E. Humphrey, J. P. Bello, and Y. Lecun. Feature Learning and Deep Architectures: New Directions for Music Informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, December 2013.

[9] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[10] Eugene Narmour. *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. University of Chicago Press, November 1992.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] 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.

[13] M. Ryynänen and A. Klapuri. Automatic Transcription of Melody, Bass Line, and Chords in Polyphonic Music. *Computer Music Journal*, 32(3):72–86, September 2008.

[14] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, Aug. 2012.

[15] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody Extraction From Polyphonic Music Signals: Approaches, Applications, and Challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[16] J. Salamon, G. Peeters, and A. Röbel. Statistical characterisation of melodic pitch contours and its application for melody extraction. In *13th Int. Soc. for Music Info. Retrieval Conf.*, pages 187–192, Porto, Portugal, Oct. 2012.

[17] J. Salamon, B. Rocha, and E. Gómez. Musical genre classification using melody features extracted from polyphonic music signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, Kyoto, Japan, Mar. 2012.

[18] H. Tachibana, T. Ono, and S. Sagayama. Melody Line Estimation in Homophonic Music Audio Signals Based on Temporal-Variability of Melodic Source. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 425–428. IEEE, 2010.