ON THE TRANSCRIPTION OF MONOPHONIC MELODIES IN AN INSTANCE-BASED PITCH CLASSIFICATION SCENARIO

Fatemeh Pishdadian, Jill K. Nelson

Department of Electrical and Computer Engineering, George Mason University

ABSTRACT

In this paper, a computationally efficient approach to transcription of monophonic melodies from a raw acoustic signal is presented. Two different instance-based pitch classification methods are proposed, the choice of which depends on the size of the available training database. In the first method, the conventional K-Nearest Neighbor algorithm is trained on a large database of piano tones and employed for monophonic pitch detection. For cases where the training database contains only one sample from each possible note, a two-step algorithm, combining semi-KNN pitch candidate selection and note sequence tracking, is suggested. It is demonstrated that in the abundance of training data, the KNN algorithm along with a proper choice of the distance measure and K, yields high performance accuracy. Furthermore, the two-step algorithm is capable of compensating for the shortage of data by incorporating prior musicological information in the transcription process.

Index Terms— Automatic music transcription, classification, K-Nearest Neighbor, piano, sequence detection

1. INTRODUCTION

Music transcription refers to the process of transforming a raw acoustic musical signal into a written representation, namely a score. Among the most fundamental pieces of information that appear in a score are the properties of individual note events, such as pitches, onset times, and durations. Once extracted, these elements can be used by a higher-level transcription system to produce note sequences, or melody lines. The work presented in this paper is focused primarily on monophonic pitch extraction and note sequence tracking in an instance-based classification scenario. In an instance-based method, no explicit assumption regarding the target function is made, and any generalization beyond the stored training samples is postponed until the classification stage.

Monophonic transcription is a relatively mature problem within music signal processing and has been studied by the research community for nearly four decades [1], and hence much of the current focus is on the polyphonic transcription problem. Monophonic transcription is still of interest, however, since single melody lines form a portion of the music corpora which is by no means negligible, e.g. single voice chants, ethnic instrumental performances, etc. Furthermore, monophonic-focused algorithms are needed, since solving the monophonic transcription problem via complex algorithms specifically designed for polyphonic scenarios is computationally inefficient. Classification-based approaches to pitch detection that make use of Artificial Neural Networks (ANNs) [2], and Support Vector Machines (SVMs) [3][4], for example, require a long and computationally demanding training stage. One of the primary shortcomings of existing work on monophonic transcription is that most of the algorithms are insufficiently evaluated. They are tested either over the sequential order of notes within a scale [5] or on a very limited range of frequencies [6]. Scales and narrow frequency ranges are not representative of actual melodic structures, and hence more sophisticated structures must be considered for meaningful algorithm evaluation.

The algorithms presented in this paper are motivated by a need for monophonic transcription techniques with significantly reduced training time, low run-time complexity, and the capability to explore melodic contours. In the design of our algorithms, we have been inspired by the probabilistic approach taken in [7], which employs the YIN algorithm [8], a modified time-domain autocorrelation method, for pitch detection and an empirical note transition probability distribution for note sequence estimation. The pitch detection in our approach, however, is performed via K-Nearest Neighbor (KNN) algorithm trained on piano tones. K-Nearest Neighbor is a low complexity method with a very simple training stage. To our knowledge, KNN has not been applied to pitch detection in the literature; autocorrelation in contrast to classification methods is by far the most common approach to single voice pitch detection [9] [10] [11]. Our experiments show that, when provided with sufficient training data, KNN can yield high performance accuracy. In the case of a very small training set (one sample per note), KNN can be combined with a sequential detection step to improve pitch sequence transcription. The combination of two steps still imposes a low computational burden compared to the aforementioned classification-based methods popular in polyphonic transcription.

2. ALGORITHM DESCRIPTION

Figure 1 provides a block diagram describing the two transcription algorithms proposed, both of which share an initial feature extraction stage. In the feature extraction stage, frequency-domain features of the note events are extracted from the input audio signal according to time-domain labels described in Section 2.1.1. If the training database is sufficiently large, a one-step algorithm (upper path) exploits these features in a classification scenario to identify the target pitch class for each note event. On the other hand, if the training database is very small, in order to improve the transcription accuracy, the transcription task is performed via a two-step algorithm (lower path). In the first step, pitch candidates for a given note event are identified in a classification scenario. In the second step, the sequential relationship between the pitch candidates is explored and the most likely note sequence, or *melody line*, is returned as the system output. Each part of the system is described in more detail in the following sections.

2.1. Feature Extraction

2.1.1. Time-domain feature extraction

The time envelope of a piano tone can be divided into four distinct sections, called attack, decay, sustain, and release (ADSR model [12]). Unlike the attack and decay sections where a broad band of frequency components are present in the tone spectrum, the fundamental frequency and its related harmonics are dominant components during the sustain section, and hence the sustain is the preferred section for frequency-domain feature extraction. Onset and sustain time detection are important elements of an overall transcription system but are outside the scope of the work presented here. In order to evaluate the performance of only the proposed note detection scheme without introducing the effects of erroneous sustain detection, we label the sustain times manually for this work.

2.1.2. Frequency-domain feature extraction

The spectrogram of an audio signal can serve as a powerful tool in joint time-frequency analysis. We consider the spectrogram of the sustain period of each event. We choose a Hamming window for spectrogram calculation to minimize leakage in the estimated spectrum. The inherent trade-off between time and frequency resolution, governed by the chosen window length, can be addressed by taking into account some of the spectral properties of musical sound. Fundamental frequencies of musical notes are distributed on a logarithmic scale, such that fundamental frequencies in lower registers are more closely spaced than those in higher registers. Therefore, time and frequency resolution demands can be better balanced in an adaptive, multi-resolution approach based on the dominant fundamental frequency range in a given excerpt of music, rather than in an approach with fixed spectrogram parameters.

The multi-resolution parameter setting is carried out within the *dominant octave detection* block. First, a rough version of the spectrogram, ($P_{I\times J}$), is generated and divided into nine segments, corresponding to the frequency ranges of seven full and two incomplete piano octaves. In the next step, total spectral power for each region within one time frame is calculated, and the region with the highest power is selected as the short-term dominant octave. Let e_m denote the range of elements of the spectrogram whose corresponding frequency components lie within the frequency range of the m^{th} piano octave ($m = 0, ..., 8$). The spectral power content in the frequency range of the m^{th} octave in the j^{th} window frame can thus be obtained from

$$P_{mj} = \sum_{i=e_m} P_{ij} \qquad j = 1, ..., J \qquad m = 0, ..., 8. \tag{1}$$

The short-term dominant octave is then given by $m_j^* = \underset{m}{\operatorname{argmax}}\ P_{mj}$, and the long-term dominant octave is given by $M^* = \underset{j}{\mathrm{MV}}\ \{m_j^*\}$, where $\underset{j}{\mathrm{MV}}\{.\}$ ($j = 1, ..., J$) indicates the *majority vote* operation over all the spectrogram columns. M^* is used as the basis for setting the time/frequency resolution, that is, the length of the window is adjusted so that the fundamental frequencies of the two lowest notes of the long-term dominant octave can be resolved. With this adaptive strategy, time resolution is compromised only when the majority of notes in a given piece of music reside in lower registers.

2.2. One-step melody transcription algorithm

K-Nearest Neighbor (KNN) is a well known instance-based classification algorithm whose training stage consists only of collecting training samples [13]. In this regard, the algorithm offers high level of simplicity compared to other adopted pitch classification methods, such as using Support Vector Machines (SVMs) to model piano tones. The KNN algorithm operates based on the assumption that each training or testing sample corresponds to a point in n-dimensional Euclidean space, $\mathbb{R}^n$. The n-dimensional feature vector representing an arbitrary sample, S, can be written as

$$S = [S(1), S(2), ..., S(n)], \tag{2}$$

where $S(n)$ denotes the value of the sample's n^{th} attribute. For the proposed piano transcription method, feature vectors are generated by extracting and averaging several consecutive columns of the spectrogram within the sustain part of each note event.

In the classification stage, the algorithm computes the distance of a given query sample from all or a local subset of training instances and assigns to the query sample the most common label (majority vote) among its K closest neighbors.

Fig. 1. Block diagram of two different melody transcription methods. The conventional KNN algorithm is employed provided that the training database is sufficiently large. If the database is of minimum size, a note sequence tracker is combined with a semi-KNN-based pitch candidate selector to compensate for the shortage of data by incorporating musicological information in the transcription process.

The maximum value of K is limited by the number of samples from each class in the training database. In the case of a minimum size training set, where only one training sample per class is available, K can be at most one.

A variety of the distance measures, including Euclidean distance (the most common), *city block* distance, and *correlation* distance, have been suggested in the KNN literature, based on specific properties of feature vectors and relative spacial distribution of classes [14]. The distance between two samples $S^{(i)}$ and $S^{(j)}$ can be calculated via the three mentioned distance measures as follows:

$$d_{eu}(S^{(i)}, S^{(j)}) = ||S^{(i)} - S^{(j)}||_2, \tag{3}$$

$$d_{cb}(S^{(i)}, S^{(j)}) = ||S^{(i)} - S^{(j)}||_1, \tag{4}$$

$$d_{crr}(S^{(i)}, S^{(j)}) = 1 - \langle S^{(i)}, S^{(j)} \rangle, \tag{5}$$

where $||.||_2$, $||.||_1$, and $\langle . \rangle$ indicate 2-norm, 1-norm, and the inner product of vectors respectively. It should be noted that in Equation (5), feature vectors are normalized so that the distance value is positive.

2.3. Two-step melody transcription algorithm

Training databases containing a large number of samples from each class might not always be available, e.g. samples from archaic instruments. For the special case in which the training database is of minimum size (one sample per class), we have developed a two-step transcription algorithm. In the first step, pitch candidates are selected in a semi-KNN scenario, and in the second step, a note sequence tracking algorithm is employed to find the most likely sequential relationship among those candidates. The reason behind adding the second step is the poor performance of the nearest neighbor pitch classifier on very small datasets. In the previous section, pitch candidates were identified independently for each note event, ignoring the correlation between successive notes in a theme. By incorporating prior information about the melodic structure we are able to compensate for the scarcity of data. The two steps of the algorithm are discussed in Sections 2.3.1 and 2.3.2.

2.3.1. Pitch candidate selection (semi-KNN)

Pitch candidates for a given note event are determined through a semi-classification-based method. Each time a note event is received, the distance between its associated feature vector and each of the training feature vectors is calculated. Then, a small number of closest training samples, which we denote by $\tilde{K}$, are recorded in a list and considered as pitch candidates for that note event. The method has a structure similar to that of the KNN algorithm in that it represents the training and testing samples by n-dimensional feature vectors and selects pitch candidates based on distance to available training samples. However, this approach cannot be regarded as a full classification procedure, since it outputs a list of candidate classes rather that a specific target class.

2.3.2. Note sequence tracking

Considering more than one pitch class at each time step ($\tilde{K} > 1$), we provide the algorithm with the capability to exploit the sequential relationship between note pitches, as well. In a maximum likelihood approach, the Viterbi algorithm [15] is applied to a trellis of pitch candidates to find the most likely note sequence. A likelihood metric is derived based on *a priori* knowledge of characteristics of note sequences and on the relationship of observed feature vectors to the training set.

The path metric expresses the likelihood of a note sequence of length L given a series of L observations, where the observations are simply feature vectors generated from the spectrogram of the music signal. We denote the note sequence and observations by $S_{1:L}$, and $S^{obs}_{1:L}$, respectively. The path metric used in the Viterbi algorithm is proportional to $P(S_{1:L}|S^{obs}_{1:L})$ and can be written as

$$\gamma(S_{1:L}) = P(S^{obs}_{1:L}|S_{1:L})P(S_{1:L}). \tag{6}$$

The observed feature vector S^{obs}_L is modeled as being drawn from a Gaussian distribution whose mean vector is

given by the training feature vector for the correct class (note). Hence, a Gaussian weight is associated with each of the $\tilde{K}$ pitch candidates according to the Euclidean distance between the $\tilde{k}^{th}$ training feature vector and the observation. We note that the choice of Gaussian distribution is made for convenience. Moreover, it facilitates the use of Euclidean distance. The use of distributions matching the properties of the other two distance measures mentioned here requires further study and will form a part of our future work.

Let $S_l^{(i)}$ denote the feature vector corresponding to the i^{th} ($i = 1, 2, ..., \tilde{K}$) pitch candidate, and S_l^{obs} denote the observed feature vector at time step l . Gaussian weights are computed from

$$P(S_l^{obs}|S_l^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{d_{eu}(S_l^{(i)}, S_l^{obs})^2}{2\sigma^2}}, \qquad (7)$$

where σ^2 is the variance of the Gaussian distribution and $d_{eu}(S^{(i)}, S^{obs})$ can be computed from (3). In general, the variance can be chosen to reflect the expected variation between training vectors and observations. In this work, it is fixed as $\sigma^2 = 1$.

To derive the *a priori* note sequence probability $P(S_{1:L})$, the note sequence or melody line is modeled as a first-order Markov process. The probability distribution over note transitions has been termed the note bigram in existing transcription literature [7]. For note transitions, the bigram $P(S_l|S_{l-1})$ is typically calculated empirically from a chosen dataset of music samples and provided as *a priori* information. In our case, it is calculated from J. S. Bach's first three Inventions (see Figure 2).

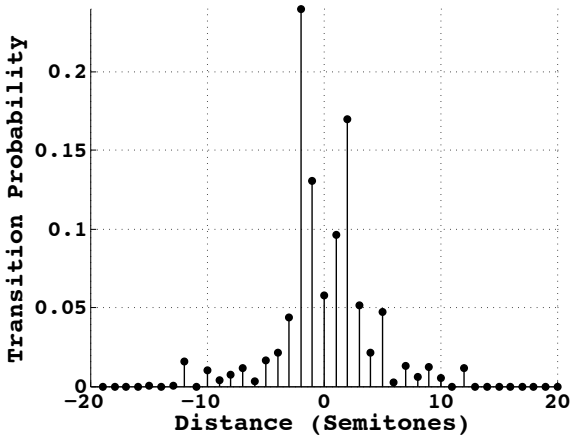

Fig. 2. Note transition probability distribution (bigram). The values on the x-axis indicate the transition distance in semitones (half notes) and on the y-axis the corresponding probabilities. For example, the probability of moving up one half step between two consecutive notes is roughly 0.1.

The total path metric for a note sequence $S_{1:L}$ of length L is computed according to

$$\gamma(S_{1:L}) = P(S_1^{obs}|S_1) \prod_{l=2}^{L} P(S_l^{obs}|S_l) P(S_l|S_{l-1}). \qquad (8)$$

The sequence of pitch classes $S_{1:L}^*$ that maximizes $\gamma(S_{1:L})$, determined via the Viterbi algorithm, is provided by the algorithm as the estimate of the note sequence, or melody line.

2.4. Computational complexity

The computational complexity of the KNN algorithm, classifying a single query sample, can be measured in terms of the number of distance calculations, which for the one-step algorithm presented here is equal to the total number of training samples, D. Thus, total computational complexity for a note sequence of length L is simply LD. The Viterbi algorithm has fixed computational complexity determined by the number of transitions in the trellis, or equivalently the number of metric calculations. For the note sequence detection problem with a first-order note transition model, the number of transitions is governed by the number of pitch candidates $\tilde{K}$ and the length of the sequence L. The total computational complexity of the two-step algorithm is the sum of complexities of both steps, $LD_{min} + \tilde{K}^2(L-1)$.

3. EXPERIMENTS

3.1. Training and testing datasets

Our training database is composed of two separate subsets, both containing recorded tones from every piano key (88 recorded tones per piano).The first subset includes tones from only one piano (Steinway grand piano - 88 samples in total). The second subset includes tones from ten different pianos (eight Steinway grand and two Steinway upright pianos - 880 samples in total). Each recorded tone is approximately 5 seconds long. The testing database includes 75 recorded monophonic themes (either right or left hand) from Bach's Inventions 1, 2, 4, and 8. All of the recordings are sampled at 44.1 kHz. We note that the testing samples are recorded on a different piano from training samples (eleven pianos in total). The lengths of the testing samples are set such that note sequences contain 22 notes on average. Training and testing feature vectors are generated from recorded note events as explained in Section 2.1.2. Because the note transition probability bigram used in these experiments was built based on the samples from Inventions 1, 2, and 3, the samples (40 in total) from Inventions 1 and 2 are treated as a validation set, and the set of themes extracted from Inventions 4 and 8 (35 in total) is treated as the true testing set.

3.2. Performance evaluation

3.2.1. One-step algorithm

The large training database (880 samples) is used to evaluate the one-step algorithm. Figure 3 provides an example of the KNN classifier output for a testing sample containing 26 note events. Notes are shown by MIDI numbers, which correspond to each piano note (half step). For reference, MIDI number 60 corresponds to Middle C (C_4 with fundamental frequency equal to 261.63 Hz). By setting $K = 3$, the algorithm classifies 25 out of 26 note pitches correctly (the second note pitch is misclassified). However, increasing the number of samples involved in the voting process to $K = 5$ degrades the performance, reducing the number of correctly classified pitches to 23 (note pitches number 2, 15, and 22 are misclassified).

The average performance results of experiments for KNN pitch classifier with three different distance measures and $K = 1, 3, 5, 7, 9$ are presented in Table 1. In order to reduce the likelihood of ties in the voting procedure, it would be preferable to consider odd numbers of neighboring points. The classification accuracy based on Euclidean and correlation distance measures peak at $K = 3$, whereas the city-block distance measure produces the best results when K is set to one. Moreover, the city-block measure outperforms the other two by at least 2%.

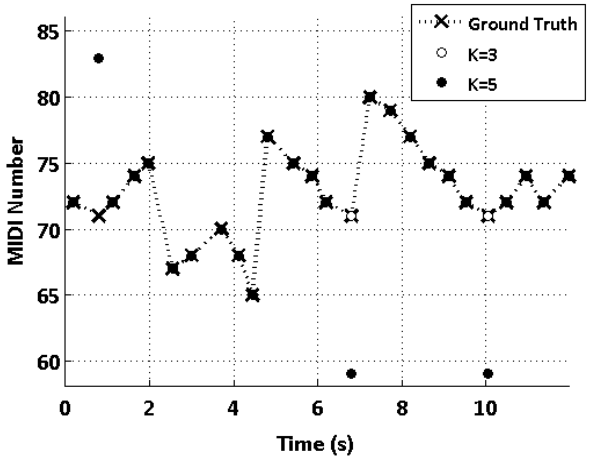

Fig. 3. An example of the KNN pitch classifier output for a sample from Invention 2 using $K = 3$ and $K = 5$.

Table 1. Performance accuracy for the KNN pitch classifier, with three different distance measures, as well as different values of K.

K	1	3	5	7	9
Euclidean	87.52	**88.87**	84.57	85.92	84.69
City block	**91.45**	88.07	85.18	86.04	85.74
Correlation	87.40	**89.36**	85.12	85.92	85.98

3.2.2. Two-step algorithm

To evaluate the performance of the two-step algorithm, the minimum-size training set (88 samples in total) is employed. As explained in Section 2.3.2, the algorithm is designed based only on the Euclidean distance and Gaussian distribution. An illustrative example of the output for a testing sample containing 23 note events is depicted in Figure 4. The algorithm makes 8 errors in detecting note event pitch with only one pitch candidate. When the number of pitch candidates is increased to two, 22 of the 23 notes are correctly classified.

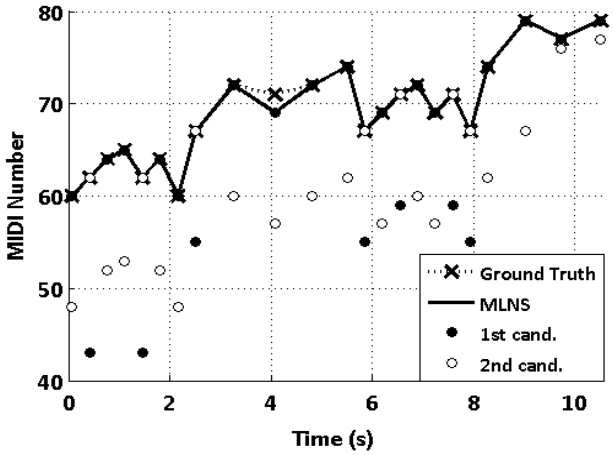

Fig. 4. An example of the two-step algorithm output for a sample from Invention 1 using $\tilde{K} = 1$ and $\tilde{K} = 2$ pitch candidates (MLNS stands for Most Likely Note Sequence).

Table 2 presents the average performance results with $\tilde{K} = 1$, 2, 3, and 4 candidates. The performance accuracy is reported for validation and testing sets both separately and combined. For any fixed number of candidates, the two sets show similar performance. It should be noted that the one-candidate case performs pitch detection based only on the observed feature vector and ignores note transition probabilities. Hence, no Viterbi search is performed in this case. Adding the second candidate improves the results by around 15%, providing a strong performance accuracy of approximately 92%. It is interesting to note that increasing the number of candidates to $\tilde{K} = 3$ and 4 does not provide any consistent performance gain.

Table 2. Note classification performance accuracy for the validation and testing sets, both separately and combined.

Number of candidates	1	2	3	4
Validation Set Acc. (%)	74.88	93.77	92.76	93.77
Testing Set Acc. (%)	76.71	90.71	91.11	90.57
Whole Dataset Acc. (%)	75.72	**92.37**	92.00	92.31

3.3. Discussion

Exploiting training data from ten different pianos along with the city block distance measure in the one-step algorithm, resulted in correctly classifying roughly 91% of the note pitches. Although the 1NN method performs poorly on the minimum-size database due to high sensitivity to noise, incorporating the spectral information and note transition probability distribution in a Viterbi search over the simplest structure of a trellis (2 pitch candidates at each time step), a performance accuracy of approximately 92% was achieved, outperforming the best result of the KNN algorithm.

Given the average length of themes, $L_{avg} = 22$, and the size of the large database, $D = 880$, the average computational complexity of the one-step algorithm can be computed as $L_{avg}D = 19360$. In the two-step algorithm, the size of the database is $D_{min} = 88$, hence the average computational complexity for $\tilde{K} = 2$ is given by $L_{avg}D_{min} + \tilde{K}^2(L_{avg} - 1) = 2020$, which shows that the second algorithm at its peak performance accuracy is around ten times less complex than the first algorithm.

4. CONCLUSIONS AND FUTURE WORK

In this paper, two algorithms for automatic transcription of monophonic melodies, both exploiting instance-based pitch classification, have been presented. The one-step method employs the KNN algorithm which works based on the distance between observed and training feature vectors and has a very short training stage composed only of collecting the training samples. The impact of KNN parameter setting, as well as the size of training database on the performance accuracy were explored. In the two-step method, a post processing stage comprised of a note sequence tracking algorithm was added to the semi-KNN-based pitch candidate selection in order to improve the performance when the size of training set is small.

Moving forward, the proposed pitch detection and melody line transcription algorithm will be applied to a broader range of instrumental and vocal timbers, as well as melodic structures from other music genres. Generalization of the bigram to other genres of music and to higher-order note dependencies will be considered. Additionally, the potential of measures other than Euclidean distance to improve the result of the two-step algorithm will be explored in further study.

5. REFERENCES

[1] Martin Piszczalski and Bernard A. Galler, "Automatic music transcription," *Computer Music Journal*, vol. 1, no. 4, November 1977.

[2] Matija Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Trans. on Multimedia*, vol. 6, no. 3, pp. 439–449, June 2004.

[3] Graham E. Poliner and Daniel P.W. Ellis, "A classification approach to melody transcription," in *Proceedings of the 6th International Conference of Music Information Retrieval*, University of London, September 2005.

[4] Graham E. Poliner and Daniel P.W. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Advances in Signal Processing*, June 2006.

[5] Mürsel Önder, Aydn Akan, and Semih Bingöl, "Pitch detection for monophonic musical notes," in *Third International Conference on Electrical and Electronic Engineering - ELECO*, December 2003, vol. 1.

[6] Juan Pablo Bello, Giuliano Monti, and Mark Sandler, "An implementation of automatic transcription of monophonic music with a blackboard system," in *Irish Signals and Systems Conference (ISSC 2000)*, June 2000.

[7] Timo Viitaniemi, Anssi Klapuri, and Antti Eronen, "A probabilistic model for the transcription of single-voice melodies," in *Proceedings of the Finnish Signal Processing Symposium*, 2003.

[8] Alain de Cheveigné and Hideki Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[9] Myron J. Ross, Harry L. Shaffer, Andrew Cohen, Richard Freudberg, and Harold J Manley, "Average magnitude difference function pitch extractor," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 22, no. 5, October 1974.

[10] "On the use of autocorrelation analysis for pitch detection," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 25, no. 1, February 1977.

[11] Giuliano Monti and Mark Sandler, "Monophonic transcription with autocorrelation," in *COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Dec. 2000.

[12] Jim Heckroth, *A Tutorial on MIDI and Wavetable Music Synthesis*, Application Note, CRYSTAL a division of CIRRUS LOGIC, 1998.

[13] Tom M. Mitchell, *MACHINE LEARNING*, The McGraw-Hill Companies, Inc., 1997.

[14] M. Marinaki, Y. Marinakis, M. Doumpos, N. Matsatsinis, and C. Zopounidis, "A comparison of several nearest neighbor classifier metrics using tabu search algorithm for the feature selection problem," *Optimization Letters*, vol. 2, no. 3, July 2007.

[15] G. David Forney, "The Viterbi algorithm," in *Proceedings of the IEEE*, March 1973, vol. 61.