# AUTOMATIC MELODY TRANSCRIPTION BASED ON CHORD TRANSCRIPTION

**Antti Laaksonen**
Department of Computer Science
University of Helsinki
ahslaaks@cs.helsinki.fi

## ABSTRACT

This paper focuses on automatic melody transcription in a situation where a chord transcription is already available. Given an excerpt of music in audio form and a chord transcription in symbolic form, the task is to create a symbolic melody transcription that consists of note onset times and pitches. We present an algorithm that divides the audio into segments based on the chord transcription, and then matches potential melody patterns to each segment. The algorithm uses chord information to favor melody patterns that are probable in the given harmony context. To evaluate the algorithm, we present a new ground truth dataset that consists of 1,5 hours of audio excerpts together with hand-made melody and chord transcriptions.

## 1. INTRODUCTION

Melody and chords have a strong connection in Western music. The purpose of this paper is to exploit this connection in automatic melody transcription. Given a chord transcription, we can use it in melody transcription to constrain the set of possible melodies. Both the rhythm and the pitches of the melody should match the chords in a successful melody transcription.

For example, let us consider the melody in Figure 1. The melody consists of 16 bars, each annotated with a chord symbol. The first observation is that chord boundaries divide the melody into segments of approximately equal length. Each of the segments has a simple rhythmical structure. In this case the chord boundaries exactly match the bar lines, and each segment contains up to three melody notes. Of course, many melodies are more complex than this, but the underlying segmentation is still usually apparent.

Let us now consider the pitches of the melody. The key of the melody mainly determines what pitches typically occur in the melody. In this example the key of the melody is C major, and almost all melody pitches belong to the C major scale. However, there are two exceptions: the G# note in the second bar and the C# note in the sixth bar. Thus, although the melody follows the C major scale, the individual chords also have an effect on the pitches. In this case the major thirds of E major and A major chords are so predominant that the melody adapts to the harmony.

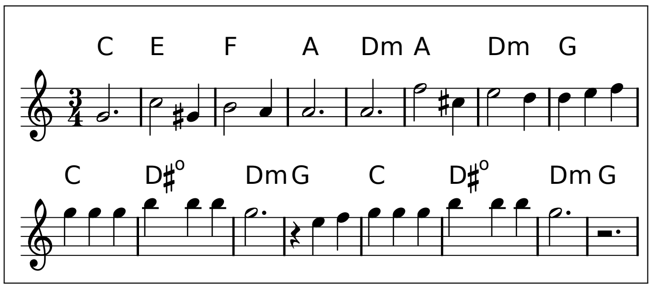


**Figure 1**: A melody from Disney's *Snow White and the Seven Dwarfs*. The chord transcription consists of 16 chords, and the melody transcription consists of 30 notes.

Human transcribers routinely use this kind of musical knowledge in music transcription. If the melody does not match the chords, or the chords do not match the melody, the transcription cannot be correct. However, in automatic music transcription, chord extraction and melody extraction have been studied separately for the most part.

Currently, the best automatic systems for chord transcription produce promising results, while melody transcription seems to be a more challenging problem. For this reason, we approach automatic melody transcription with the assumption that a chord transcription has already been done. We present an algorithm that divides the audio data into segments based on the chord boundaries. After this, the algorithm assigns each segment a melody pattern that matches both the audio data and the chord information.

### 1.1 Problem statement

Given an excerpt of music in audio form and a chord transcription in symbolic form, the task is to produce a melody transcription in symbolic form. We concentrate on typical Western music, and assume that the pitches of the notes are given in semitones.

We assume that the audio data $A$ is divided into $n_A$ frames of equal length using some preprocessing method. For each audio frame $k$ ($1 \leq k \leq n_A$), we are given values $A[k].begin$ and $A[k].end$ that are time values in seconds when the frame begins and ends. In addition, for each possible melody note $q$ we are given a real number $A[k][q]$ in the range $[0, 1]$. This value estimates the strength of the note $q$ in frame $k$.

The chord transcription $C$ consists of $n_C$ chord changes. For each chord change $k$ ($1 \leq k \leq n_C$), we are given a value $C[k].time$ that is the time when the chord changes. In addition, we are given a value $C[k].chord$ that is the name of the chord. We restrict ourselves to triads (major, minor, diminished and augmented chords that consist of three notes), which results in a total of 48 possible chords.

Finally, the outcome of the algorithm should be a melody transcription $M$ that consists of $n_M$ melody notes. For each melody note $k$ ($1 \leq k \leq n_M$), the algorithm should produce values $M[k].time$ and $M[k].pitch$ that denote the onset time of the note and the pitch of the note.

Throughout the paper, we use MIDI note numbers to refer to the pitches. Thus, every pitch has a unique integer value and the interval of pitches $a$ and $b$ is $|a-b|$ semitones. Pitch C4 (261.6 Hz) is associated with MIDI value 60.

### 1.2 Related work

Automatic melody transcription has been studied actively during the last decade. Detailed reviews of the proposed methods can be found in [16] and [20].

The usual first step in automatic melody transcription is to detect potential melody notes in the audio signal. The most popular method for this is to calculate a salience function for the audio frames using the discrete Fourier transform or a similar technique (e.g. [2, 6, 15, 19]). Other proposed approaches for audio data processing include signal source separation [3] and audio frame classification [5].

After this, the final melody is selected according to some criterion. One technique for this is to construct a hidden Markov model (HMM) for note transitions and use the Viterbi algorithm for tracking the most probable melodic line [3, 5, 19]. An alternative to this is to use a set of local rules that describe typical properties of melody notes and outlier notes [7, 15, 20]. In addition, some systems [2, 6] feature agents that follow potential melody lines.

The idea of providing additional information to facilitate the melody transcription has also been considered in previous studies. A usual approach for this is to gather information from users. For example, users can determine which instruments are present [8], help in the source separation process [4] or create an initial version of the transcription [9]. The drawback of these systems is, of course, that the transcription is not fully automatic.

There are also some previous studies that combine key, chord and pitch estimation. In [18], the key and the chords of the piece are estimated simultaneously. Multiple pitch transcription systems that exploit key and chord information in pitch estimation include [1] and [17].

Most of the previous work on automatic melody transcription focuses on a slightly different problem than the topic of this paper, namely how to determine the melody frequency in the audio signal frame-by-frame. In [19] and [22], the output of the algorithm is similar to ours.

## 2. ALGORITHM

In this section we present our melody transcription algorithm that uses a chord transcription as a starting point for the transcription. The algorithm first divides the audio data into segments so that the boundaries of the segments correspond to the boundaries of the chords in the chord transcription. After this, the key of the piece is estimated using the chord transcription. Finally, the algorithm assigns each segment a pattern of notes that matches both the audio data and the chord transcription.

The input and the output of the algorithm are as described in Section 1.1. Thus, the algorithm is given $n_A$ audio frames in array $A$ and a chord transcription of $n_C$ chord changes in array $C$, and the algorithm produces a melody transcription of $n_M$ notes in array $M$.

### 2.1 Segmentation

The first step in the algorithm is to divide the audio data into segments. The segments will be processed separately in a later phase in the algorithm. The idea is to choose the boundaries of the segments so that the harmony in each segment is stable. This is accomplished using the chord boundaries in the chord transcription.

Let

$$l(k) = C[k+1].time - C[k].time$$

for each $k$ where $1 \leq k \leq n_C - 1$ and

$$f(k, x) = (l(k)/x)/\lfloor l(k)/x \rfloor$$

where $x$ is a real value. Thus, $l(k)$ is the length of the segment between chord changes $k$ and $k+1$, and $f(k, x)$ is an estimate how evenly $x$ divides that segment into smaller segments. Finally, let

$$g(k, x) = \begin{cases} 1 & \text{if } f(k, x) \leq 1 + \epsilon \\ 0 & \text{otherwise} \end{cases}$$

and

$$s(x) = \sum_{i=1}^{n_C - 1} g(i, x).$$

If $f(k, x) \leq 1 + \epsilon$ for some small $\epsilon$, our interpretation is that $x$ divides the segment evenly. Thus, $g(k, x)$ indicates if the segment is divided evenly, and $s(x)$ is the number of segments that are divided evenly if $x$ was chosen. In this paper we use value $\epsilon = 0.1$.

The algorithm chooses a value of $x$ such that $x$ is in the range $[min_x, max_x]$ and $s(x)$ is as large as possible. The value $x$ will be used as a unit length in the segmentation. The values $min_x$ and $max_x$ denote the minimum and maximum unit length; in this paper we use values $min_x = 0.5$ and $max_x = 3$.

Finally, the algorithm produces a segment division $S$ of $n_S$ segments by dividing each chord segment $k$ into $l(k)/x$ smaller segments of equal length (the number of segments is rounded to the nearest integer). For each new segment $u$ ($1 \leq u \leq n_S$) the algorithm assigns the values $S[u].begin$ and $S[u].end$ as described above, and $S[u].chord$ denotes the name of the chord in the segment.

### 2.2 Key estimation

After determining the segments, the algorithm estimates the key of the piece. The estimated key will be used later in the algorithm to favor melody notes that agree with the key. The key estimation is done using a simple method that is based on the chord information.

The algorithm goes through all segments in $S$ and maintains a counter for each pitch class (a total of 12 counters). Initially, all the counters are zero. For each segment $k$, the algorithm increases the value of each counter that corresponds to $S[k].chord$. For example, if $S[k].chord$ is G major, the algorithm increases counters that correspond to notes G, B and D.

Finally, the algorithm determines the key using the counters as follows. There are 24 possible keys, 12 major keys and 12 minor keys. For each key, the algorithm calculates the sum of counters that correspond to tonic, mediant and dominant in that key. The key whose sum is the highest is selected as the key of the piece.

This method for key estimation is more simple than methods used in previous studies involving chord and key estimation from music audio [11,18]. However, this method produces results that are considered accurate enough for this purpose.

### 2.3 Pattern matching

For each segment, the algorithm chooses a melody pattern that matches both the audio data within the segment and the chord and key information. Each segment is processed independently, and the final melody transcription consists of all melody patterns in the segments.

The algorithm divides each segment into $d$ note slots where $d$ is a preselected constant for all segments. Each note slot can contain either one melody note or rest in the melody pattern. The idea is to select $d$ so that most rhythms can be represented using $d$ note slots, but at the same time $d$ is small enough to restrict the number of melody notes. In practice, small integers that are divisible by 2 and/or 3 should be good choices for $d$.

An optimal melody pattern is selected according to a scoring function. The scoring function should give high scores for melody patterns that are probable choices for the segment. Depending on the scoring function, there are three ways to construct the optimal melody pattern:

- Greedy construction: If the melody slots are independent of one another, we can select the optimal melody note for each slot and combine the results to get the optimal melody pattern.
- Dynamic programming: If the melody slots are not independent but the score of a slot only depends on the previous slot, we can use dynamic programming to construct the optimal pattern.
- Complete search: If the score of a melody pattern cannot be calculated before all melody notes are selected, we have to go through all possible note patterns and select the optimal one.

The methods involving greedy construction and dynamic programming are efficient in all practical situations. However, in complete search we need to check $q^d$ melody patterns where $q$ is the number of possible choices for a melody slot. In practice, $q \approx 50$, so complete search can be used only when $d \leq 4$ to keep the algorithm efficient.

### 2.4 Scoring function

The scoring function that we use in this paper is primarily based on the key information and favors melody notes that match the estimated key of the excerpt. In addition, the notes that belong to the chord of the segment have an increased probability to be selected to the melody. Thus, if an E major chord occurs in the C major key, the note G# is a strong candidate for the melody note even if it does not belong to the C major scale.

Let $s(k, a, q)$ denote the score for an event where $a$'th note slot of segment $k$ contains note $q$. We calculate the score using the formula

$$s(k, a, q) = z \cdot b(k, a, q) - x \cdot c(k, a, q)$$

where $b(k, a, q)$ is a base score calculated from the audio data, $c(k, a, q)$ is a penalty for selecting a note that does not appear in the audio data, and $z$ and $x$ are parameters that control the balance of the base score and the penalty.

Let $I$ be a set that contains the indices of all audio frames that are inside the current note slot. Now we define

$$b(k, a, q) = \sum_{i \in I} A[i][q]$$

and

$$c(k, a, q) = \sum_{i \in I} e(i, q)$$

where

$$e(i, q) = \begin{cases} 1 & \text{if } A[i][q] = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The parameter $z$ favors melody notes that match the chord transcription, and it should depend on the key of the excerpt and the chord in segment $k$. We set $z = 2$ if $q$ belongs to the current chord, $z = 1$ if $q$ belongs to the key of the excerpt and otherwise $z = 0$. The parameter $x$ controls the effect of adding a note to the melody that does not appear strongly in the audio data, and we study the effect of that constant in Section 3.

Finally, we select the note pattern greedily so that each note slot will be assigned the note that maximizes the score for that slot. If no note produces a score greater than 0, we leave that slot empty.

We also experimented with dynamic programming scoring functions that favor small intervals between consecutive notes, but the results remained almost unchanged. Consequently, we chose the more simple greedy construction approach.

## 3. EVALUATION

In this section we present results concerning the accuracy of the transcriptions produced by our algorithm, using real-world inputs. We evaluated our algorithm using a dataset of Western popular music. We used both hand-made and automatic chord transcriptions as additional input for the algorithm.

Audio Melody Extraction task is an established part of the MIREX evaluation [13]. However, in the MIREX evaluation each audio frame is assigned a melody note frequency, and those results cannot be compared with our melody transcriptions that consist of note onset times and pitches in semitones.

### 3.1 Dataset

The evaluation dataset consists of 1,5 hours of audio excerpts from Western popular music. The length of each excerpt in the dataset is between 20 and 60 seconds. For each excerpt, we manually created time-aligned melody and chord transcriptions. We chose the excerpts so that the content of each excerpt is unique i.e. repetitions of verses and choruses are not included in the dataset.

The dataset can be found on our web site [1] , and is available for free for use in research. For each excerpt, the dataset includes an audio file in WAV format, and chord and melody transcriptions in text format. Each chord transcription is a list of chord change times and chord symbols, and each melody transcription is a list of note onset times and pitches. Thus, the transcriptions in the dataset correspond to the definitions in Section 1.1.

### 3.2 Evaluation method

To evaluate a melody transcription, we calculate two values: the precision and the recall. Precision is the ratio of the number of correct notes in the transcription and the total number of notes in the transcription. Recall is the ratio of the number of correct notes in the transcription and the total number of notes in the ground truth.

Let $X$ be a melody transcription of $n_X$ notes created by the algorithm, and let $G$ be the corresponding melody transcription of $n_G$ notes in the ground truth. Both transcriptions consists of a list of melody note onset times and pitches as described in Section 1.1.

To evaluate the precision and the recall of $X$, we first align the transcriptions. Let $n_D$ be the maximum integer value such that we can create lists $D_X$ and $D_G$ as follows. List $D_X$ consists of $n_D$ note indices in $X$, and list $D_G$ consists of $n_D$ note indices in $G$. In addition, for each $k$ ($1 \le k \le n_D$) $X[D_X[k]].pitch = G[D_G[k]].pitch$ and $|X[D_X[k]].time - G[D_G[k]].time| \le \alpha$ where $\alpha$ is a small contant. In other words, we require that lists $D_X$ and $D_G$ align a set of notes where all pitches match each other and the onset times of the notes do not differ more than $\alpha$ from each other.

Finally, let

$$\text{precision}(X, G) = n_D/n_X$$

and

$$\text{recall}(X, G) = n_D/n_G.$$

[1] `http://cs.helsinki.fi/u/ahslaaks/fpds/`

In practice, we calculate the value $n_D$ efficiently using dynamic programming. The technique is similar to calculating the Levenshtein distance between two strings [14].

This evaluation method corresponds with that used in [19] and [22], however, the previous papers do not specify how the notes in the two transcriptions are aligned.

### 3.3 Experiments

We implemented our algorithm as described in Section 2. For calculating array $A$ we used an algorithm by Salamon and Gómez that estimates potential melody contours in the audio signal. We used the Vamp plugin implementation of the algorithm (”all pitch contours”). Note that this algorithm already restricts the set of possible melodies considerably. We converted each pitch frequency to a MIDI note number assuming that the frequency of A4 is 440 Hz.

We used four chord transcriptions in the evaluation:

- A random chord transcription where the time between two chord changes is a random real number in the range $[0.5, 2]$ and each chord is randomly selected from the set of 48 possible triads.
- A simple automatic chord transcription created by our own algorithm. We used the standard technique of constructing a hidden Markov model and tracking the optimal path using the Viterbi algorithm [21].
- An advanced automatic chord transcription created using the Chordino tool [12].
- The chord transcription in the ground truth.

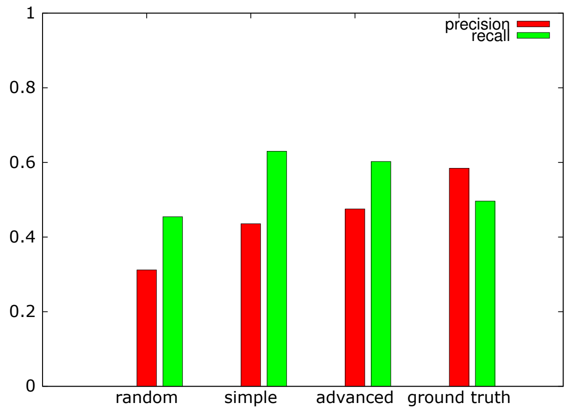
Random chord transcriptions were used in an effort to understand the actual role of the chord information and how the algorithm works if the chord information does not make sense at all.

Finally, there are three parameters that we varied during the evaluation:

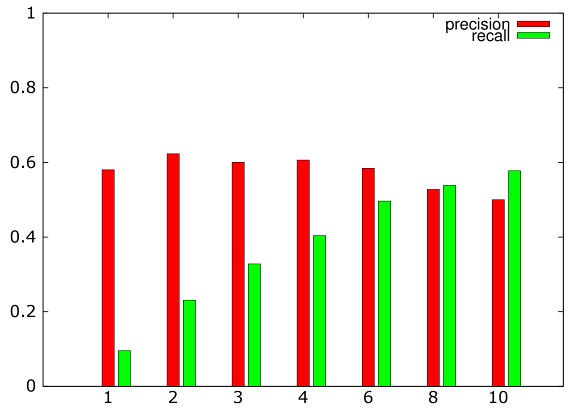
- $d$: the number of note slots in a segment as described in Section 2.3 (default value: 6),
- $x$: the cost for assigning a note to a frame without a note as described in Section 2.4 (default value: 1.00),
- $\alpha$: the maximum onset time difference in the evaluation as described in Section 3.2 (default value: 0.25).

The default values were chosen so that they produce good results on the evaluation dataset.

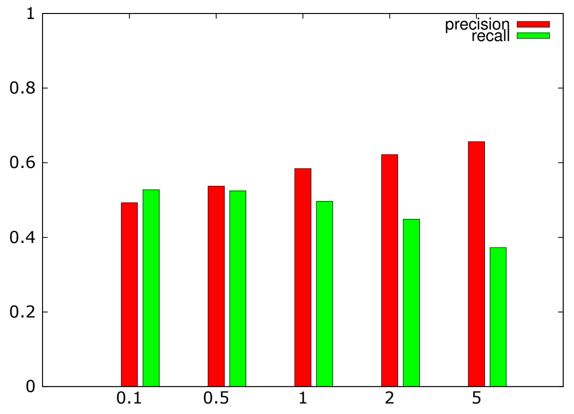
In each experiment in the evaluation, we varied one parameter and kept the remaining parameters unchanged. We used the ground truth chord transcription as the default chord transcription. We created melody transcriptions for all excerpts in the dataset and calculated average precision and recall values.



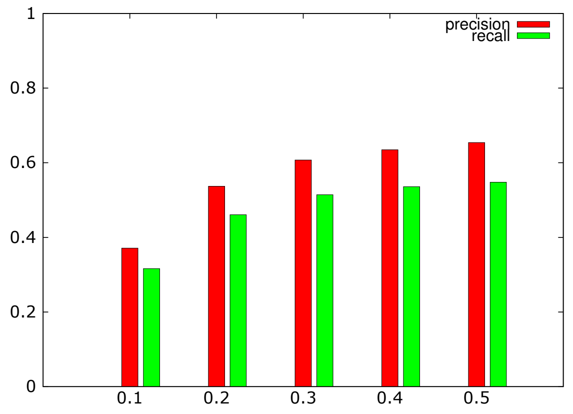
(a) The results using random chord transcription, simple automatic transcription, advanced automatic transcription, and ground truth transcription.



(b) The results varying the parameter $d$: the number of available note slots in a segment.



(c) The results varying the parameter $x$: the cost for assigning a note to a frame without a note.



(d) The results varying the parameter $\alpha$: the maximum onset time difference in seconds in the evaluation.

**Figure 2**: The results of the experiment.

### 3.4 Results

In the first experiment (Figure 2a) we studied how the quality of the chord transcription affects the results. As expected, the better the chord transcription, the better the precision of the melody transcription. However, recall was highest when using automatic chord transcriptions. One possible reason for this is that there were more chord changes in automatic transcriptions than in the ground truth transcription. Therefore more melody notes were selected using the automatic chord transcriptions.

In the second experiment (Figure 2b) we varied the parameter $d$: the number of note slots in a segment. Our findings suggest that 6 note slots is a good trade-off between the precision and the recall. This can be explained by the fact that 6 is divisible by both 2 and 3, and thus segments of 6 note slots are suitable for both 3/4 time and 4/4 time music. Interestingly, when $d \leq 6$ the precision of the transcription remained nearly unchanged.

In the third experiment (Figure 2c) we varied the parameter $x$: the cost for assigning a note to a frame without a note. This was an important parameter, and the results were as expected. Increasing the parameter $x$ improves the precision because melody notes are only selected if they appear strongly in the audio data. At the same time, this decreases the recall because fewer uncertain notes are included in the melody transcription.

Finally, in the fourth experiment (Figure 2d) we varied the parameter $\alpha$: the maximum note onset time difference in the evaluation. Of course, the greater the parameter $\alpha$, the better the results. Interestingly, after reaching a value of approximately 0.25, increasing $\alpha$ did not affect the results considerably. The probable reason for this is that if the melody note pitches in the transcription are not correct, the situation cannot be rescued by allowing more error for the onset times.

Previous studies also present some results about the precision and the accuracy of the algorithms. However, findings from these studies cannot be directly compared with the new results because the evaluation dataset is different in each study. In [19] precision 0.49 and recall 0.61 was reported using a database of 84 popular songs. In [22] the melody transcription was evaluated using a small set of 11 songs with precision 0.68 and recall 0.63.

## 4. CONCLUSIONS

In this paper we presented an automatic melody transcription algorithm that uses a chord transcription for selecting melodies that match the harmony of the music. We evaluated the algorithm using a collection of popular music excerpts, and the results of the evaluation suggest that the chord information can be successfully used in melody transcription of real-world inputs.

Our new evaluation dataset consists of 1,5 hours of audio excerpts of popular music together with melody and chord annotations. The dataset can be used at no cost for research purposes, for example as evaluation material for other chord transcription and melody transcription systems.

Our future work aims to use the harmony information provided by the chord transcription more extensively in melody transcription. Currently our algorithm uses only information about chord notes to constrain the pitches of melody notes, but using more advanced musical knowledge should yield better results.

## 5. ACKNOWLEDGEMENTS

This work has been supported by the Helsinki Doctoral Programme in Computer Science and the Academy of Finland (grant number 118653).

## 6. REFERENCES

[1] E. Benetos, A. Jansson and T. Weyde: ”Improving automatic music transcription through key detection,” *AES 53rd International Conference on Semantic Audio*, 2014.

[2] K. Dressler: “An auditory streaming approach for melody extraction from polyphonic music,” *12th International Society for Music Information Retrieval Conference*, 19–24, 2011.

[3] J.-L. Durrieu et al: ”Source/filter model for unsupervised main melody extraction from polyphonic audio signals,” *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3), 564–575, 2010.

[4] J.-L. Durrieu and J.-P. Thiran: ”Musical audio source separation based on user-selected F0 track,” *10th International Conference on Latent Variable Analysis and Signal Separation*, 2012.

[5] D. Ellis and G. Poliner: “Classification-based melody transcription,” *Machine Learning*, 65(2–3), 439–456, 2006.

[6] 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, 2004.

[7] S. Joo, S. Park, S. Jo and C. Yoo: “Melody extraction based on harmonic coded structure,” *12th International Society for Music Information Retrieval Conference*, 227–232, 2011.

[8] H. Kirchhoff, S. Dixon and A. Klapuri: ”Shift-variant non-negative matrix deconvolution for music transcription,” *37th International Conference on Acoustics, Speech and Signal Processing*, 2012.

[9] A. Laaksonen: ”Semi-automatic melody extraction using note onset time and pitch information from users,” *SMC Sound and Music Computing Conference*, 689–694, 2013.

[10] M. Lagrange et al: ”Normalized cuts for predominant melodic source separation,” *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2), 278–290, 2008.

[11] K. Lee and M. Slaney: ”A unified system for chord transcription and key extraction using hidden Markov models,” *8th International Conference on Music Information Retriavel*, 245–250, 2007

[12] M. Mauch and S. Dixon: ”Approximate note transcription for the improved identification of difficult chords,” *11th International Society for Music Information Retrieval Conference*, 135–140, 2010

[13] MIREX Wiki: Audio Melody Extraction task, http://www.music-ir.org/mirex/wiki/

[14] G. Navarro: ”A guided tour to approximate string matching,” *ACM Computing Surveys*, 33(1): 31–88, 2001

[15] R. Paiva, T. Mendes and A. Cardoso: ”Melody detection in polyphonic musical signals: exploiting perceptual rules, note salience, and melodic smoothness,” *Computer Music Journal*, 30(4), 80–98, 2006.

[16] G. Poliner et al: ”Melody transcription from music audio: approaches and evaluation,” *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4), 1247–1256, 2007.

[17] S. Raczyński, E. Vincent, F. Bimbot and S. Sagayama: ”Multiple pitch transcription using DBN-based musicological models,” *11th International Society for Music Information Retrieval Conference*, 363–368, 2010

[18] T. Rocher et al: ”Concurrent estimation of chords and keys from audio,” *11th International Society for Music Information Retrieval Conference*, 141–146, 2010

[19] M. Ryynänen and A. Klapuri: ”Automatic transcription of melody, bass line, and chords in polyphonic music,” *Computer Music Journal*, 32(3), 72–86, 2008.

[20] 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, 2012.

[21] A. Sheh and D. Ellis: “Chord segmentation and recognition using EM-trained hidden Markov models,” *4th International Conference on Music Information Retrieval*, 183–189, 2003

[22] J. Weil et al: ”Automatic generation of lead sheets from polyphonic music signals,” *10th International Society for Music Information Retrieval Conference*, 603–608, 2009