TRANSCRIBING HUMAN PIANO PERFORMANCES INTO MUSIC NOTATION

Andrea Cogliati[*], David Temperley[+], Zhiyao Duan[*]
Electrical and Computer Engineering[*] and Eastman School of Music[+], University of Rochester
{andrea.cogliati,zhiyao.duan}@rochester.edu
dtemperley@esm.rochester.edu

ABSTRACT

Automatic music transcription aims to transcribe musical performances into music notation. However, existing transcription systems that have been described in research papers typically focus on multi-F0 estimation from audio and only output notes in absolute terms, showing frequency and absolute time (a piano-roll representation), but not in musical terms, with spelling distinctions (e.g., A♭ versus G♯) and quantized meter. To complete the transcription process, one would need to convert the piano-roll representation into a properly formatted and musically meaningful musical score. This process is non-trivial and largely unresearched. In this paper we present a system that generates music notation output from human-recorded MIDI performances of piano music. We show that the correct estimation of the meter, harmony and streams in a piano performance provides a solid foundation to produce a properly formatted score. In a blind evaluation by professional music theorists, the proposed method outperforms two commercial programs and an open source program in terms of pitch notation and rhythmic notation, and ties for the top in terms of overall voicing and staff placement.

1. INTRODUCTION

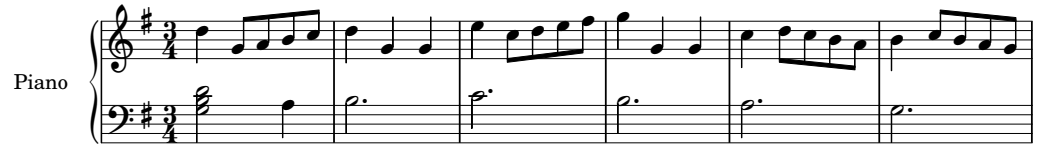
Automatic Music Transcription (AMT) is the process of inferring a symbolic music representation from a music performance, such as a live performance or a recording. The output of AMT can be a full musical score or an intermediate representation, such as a MIDI file [5]. AMT has several applications in music education (e.g., providing feedback to piano students), content-based music search (e.g., searching for songs with a similar chord progression or bassline), musicological analysis of non-notated music (e.g., jazz improvisations and most non-Western music), and music enjoyment (e.g., visual representation of the music content).

Intermediate representations are closer to the audio file even though they identify certain kinds of musical information that are not readily accessible, such as explicit pitches and note onsets. However, the encoding of this information is generally not done in abstract musical terms but still reflects some of the arbitrariness of a human performance; e.g., note onsets may be expressed in terms of absolute times instead of being quantized to a meter, and pitches may be expressed in terms of frequency or MIDI note numbers, instead of proper note spelling, e.g., C♯ has the same MIDI note number as D♭. Music notation provides further information such as key signature, time signature, rhythmic values, barlines, and voicing (e.g., the representation of multiple voices with upward and downward stems); this information is useful and indeed virtually necessary for further performance and analysis [13].

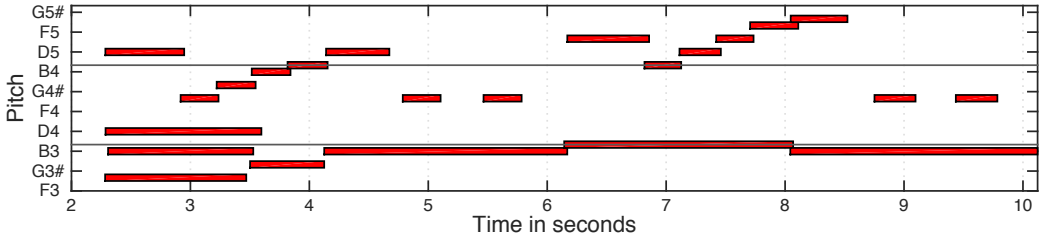
While AMT was initially formulated as a method to convert musical sounds into common music notation [15], most AMT systems so far have opted for lower level representations [5]; very few systems have attempted to estimate higher level musical information, such as beats or pattern repetitions, directly from the audio [9, 14]. Higher level musical information can also be estimated from an intermediate representation [6, 18]. In this paper we opt for the latter approach; this allows the conversion of MIDI to notation, and eventually (in combination with an audio-to-MIDI conversion system, such as [8]) could generate notation from audio as well.

A MIDI file can represent a piano performance very accurately; in fact, the only variables involved are note onset, offset, velocity and pedal activation. Moreover, MIDI representations of piano performances can be recorded from a MIDI keyboard, or from a piano with key sensors. The MIDI standard is capable of encoding high-level musical information, such as key and time signatures, into MIDI files, but this information is not typically included in recorded performances, unless the performer manually inserts it. Furthermore, recorded MIDI performances are typically unquantized, as performers continuously change the speed of playing to obtain a more expressive performance, and may play certain notes slightly earlier or later than they should be to highlight certain musical lines.

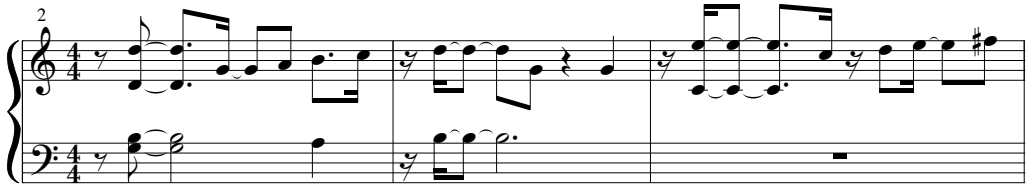
The process of producing a correct full music notation from an unquantized and un-annotated MIDI file is non-trivial and, to the authors' knowledge, no system capable of producing full music notation has been implemented and documented in academic research papers thus far. Without a proper estimation of the meter and the har-



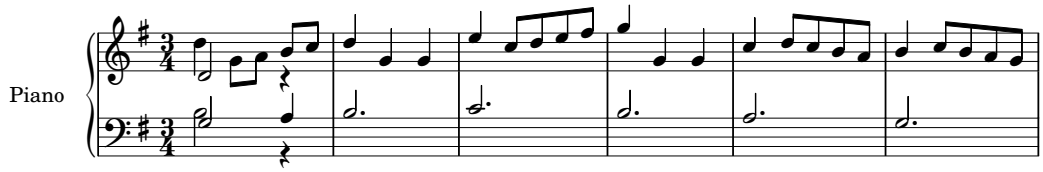
(a) Ground Truth



(b) Unquantized MIDI piano roll



(c) GarageBand



(d) Proposed method

Figure 1. Transcription of a performance of the Minuet in G from Bach's Notebook for Anna Magdalena Bach. (a) shows the original score (b) shows the unquantized pianoroll of a MIDI performance. (c) shows the output from GarageBand, which does not perform any analysis on the MIDI file. (d) shows the output of the proposed method after estimating the correct meter, key signature, beats and streams. The music excerpts are of different lengths for better formatting.

mony, the results are very poor – see Fig. 1 (c). The task can be divided into two main sub-tasks: musical structure analysis and note placement on the score. For the first sub-task, the MIDI file must be analyzed to estimate the key signature and the correct note spelling, as well as the beats and the correct time signature. For the second sub-task, once the notes have been correctly spelled and quantized to the correct meter, they must be properly positioned on the staff. Piano music is normally notated on two staves. The higher staff is usually notated in treble clef, and contains the notes generally played by the right hand. The lower staff is usually notated in bass clef, and contains the notes generally played by the left hand. Notes should be placed on staves to simplify the reading of the score, e.g., notes should be well spaced and typographical elements should not clash with each other. The placement of the notes and other typographical elements also convey musical meanings, e.g., notes pertaining to the same voice should have the stems pointing in the same direction and beaming should follow the rhythm of the musical passage. Finally, concurrent notes played by a single hand as chords should share the same stem. Exceptions to these basic rules are not uncommon, typically to simplify the reading by a performer, e.g., if a passage requires both hands to play in the higher range of the piano keyboard, both staves may be notated in the treble clef to avoid too many ledger lines and too many notes on the same staff.

In this paper we present a novel method to fully notate a piano performance recorded as an unquantized and un-annotated MIDI file, in which only the note pitches (MIDI number), onsets and offsets are considered. The initial analysis of the piece is done through a probabilistic model proposed by Temperley to jointly estimate meter, harmony and streams [18]. The engraving of the score is done through the free software LilyPond [1]. The evaluation dataset and the Python code are available on the first author's web site [1] .

[1] http://www.ece.rochester.edu/~acogliat/

2. RELATED WORKS

There are several free and commercial programs, such as Finale, Sibelius and MuseScore, that can import MIDI files and translate them into full music notation, but they typically require user intervention to inform the process to a certain degree. For instance, Finale requires the user to manually select the time signature, while it can infer the key signature from the file itself. Certain sequencers and Digital Audio Workstations, such as GarageBand and Logic Pro, have various functions to facilitate the import of MIDI files; for example, Logic Pro has a function to align the time track to the beats in the MIDI files, but requires the user to input the time signature and estimate the initial tempo of the piece.

Among the programs used for the evaluation of the proposed method, MuseScore [2] has the most advanced MIDI file import feature. MuseScore has a specific option to import human performances, and is capable of estimating the meter and the key signature. During the experiment, MuseScore showed a sophisticated capability to position different voices on the piano staves, which resulted in high scores from the evaluators, especially in terms of overall voicing and staff placement. Unfortunately, details on how all these steps are performed are not documented in the website [2] and have not been published in research papers.

The task of identifying musical structures from a MIDI performance has been extensively researched, especially in the past two decades. Cambouropoulos [6] describes the key components necessary to convert a MIDI performance into musical notation: identification of elementary musical objects (i.e., chords, arpeggiated chords, and trills), beat identification and tracking, time quantization and pitch spelling. However, the article does not describe how to render a musical score from the modules presented. Takeda et al. [16] describe a Hidden Markov Model for the automatic transcription of monophonic MIDI performances. In his PhD thesis, Cemgil [7] presents a Bayesian framework for music transcription, identifying some issues related to automatic music typesetting (i.e., the automatic rendering of a musical score from a symbolic representation), in particular, tempo quantization, and chord and melody identification. Karydis et al. [12] proposes a perceptually motivated model for voice separation capable of grouping polyphonic groups of notes, such as chords or other forms of accompaniment figures, into a perceptual stream. A more recent paper by Grohganz et al. [11] introduces the concepts of score-informed MIDI file (S-MIDI), in which musical tempo and beats are properly represented, and performed MIDI file (P-MIDI), which records a performance in absolute time. The paper also presents a procedure to approximate an S-MIDI file from a P-MIDI file – that is, to detect the beats and the meter implied in the P-MIDI file, starting from a tempogram then analyzing the beat inconsistency with a salience function based on autocorrelation.

Musical structures can also be inferred directly from audio. Ochiai et al. [14] propose a model for the joint estimation of note pitches, onsets, offsets and beats based on Non-negative Matrix Factorization constrained with a rhythmic structure modeled with a Gaussian mixture model. Collins et al. [9] propose a model for multiple F0 estimation, beat tracking, quantization, and pattern discovery. The pitches are estimated with a neural network. A Hidden Markov Model (HMM) is separately used for beat tracking. The results are then combined to quantize the notes. Note spelling is performed by estimating the key of the piece and assigning to MIDI notes the most probable pitch class given the key.

3. PROPOSED METHOD

The proposed method takes an unquantized and unannotated MIDI file as input. The following subsections explain each step in the proposed method. The entire process is illustrated in Fig. 2. An example of the output is shown in Fig. 1 (d).

3.1 Fix spurious overlapping notes

The first step is to fix spurious overlapping notes. Piano players do not play notes with the correct length all the time. As we can see from Fig. 1 (b), certain notes are played shorter than they should be, resulting in gaps between notes, while other notes are played longer than they should be, resulting in overlapping notes. Gaps between notes in the same melodic line might result in extra rests in the score, while overlapping notes might result in extra streams being created by the probabilistic model [18] used in the next step, resulting in extra voices in the final score. In particular, the probabilistic model used in this paper always assigns overlapping notes to different streams, so it is critical to remove erroneous overlaps.

To estimate whether the overlap is correct or wrong we consider pairs of overlapping notes separately. For each pair, we calculate one overlapping ratio for each note. The ratio is defined as the length of the overlapping region over the length of the note. The overlap is considered spurious if the sum of the two ratios is below a certain threshold. For the experiment we set a threshold of 30%. The output of the first step is a note list, i.e., a list of note events, each including an onset, a duration (both in milliseconds), and a MIDI note number. An example is shown in Fig. 3. Notice the small overlaps in the top figure between the three low notes in the initial chord and the second bass note, as well as the short overlaps in the scale in the soprano line; these are removed in the second figure. Also notice that correct overlapping notes, such as a melody line moving over the same bass note, are preserved.

3.2 Estimate meter, harmony and streams

In the second step, we apply the probabilistic model [18] to the note list. The probabilistic model estimates the meter, the harmony, and the streams. The meter and harmony are estimated in a single joint process. This process is modeled as an HMM and is based on the concept of tactus-root combination (TRC), a combination of two adjacent tactus

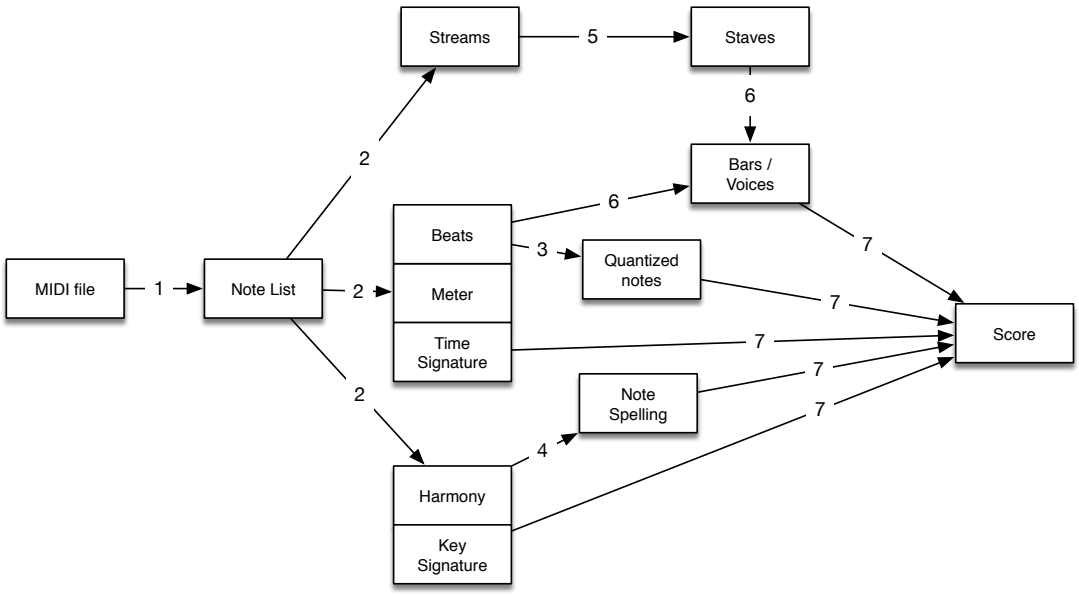
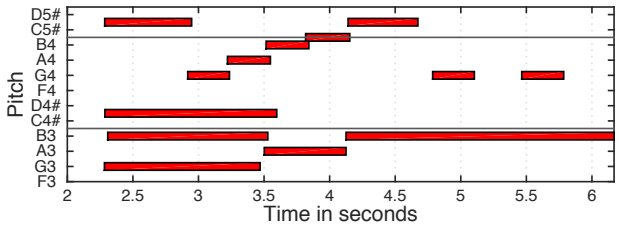
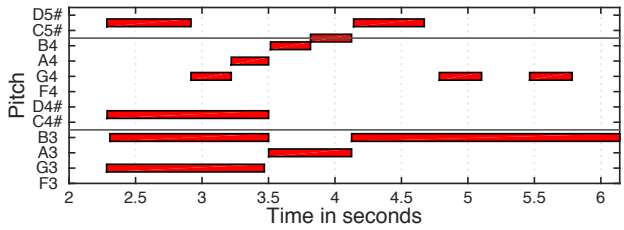


Figure 2. Illustration of the proposed method. The arrows indicate dependencies between entities. The numbers refer to the steps (subsection numbers) in Section 3.



(a) Original pianoroll of a MIDI performance



(b) Pianoroll after fixing spurious overlapping notes

Figure 3. An example of the step of fixing spurious overlapping notes.

beats and a chord root. The probability of a TRC only depends on the previous TRC, and the probability of beats and notes within a TRC only depends on the TRC. The musical intuition behind this is that the "goodness" of a tactus interval depends only on its relationship to the previous tactus interval (with a preference to minimize changes in length from one interval to the next), the goodness of a root depends only on the previous root (with a preference to maintain the same root if possible, or to move to another root that is a fifth away), and the goodness of a particular pattern of notes within a short time interval depends only on the current root and the placement of beats within that interval (with a preference for note onsets on tactus beats or at plausible points–e.g., roughly halfway–in between them, and a preference for notes that are chord-tones of the root). The process also considers different divisions of the tactus interval (representing simple or compound meter) and placements of strong beats (duple versus triple meter). In the current context, the metrical analysis is useful for the placement of barlines and for rhythmic notation; the harmonic analysis is useful for pitch spelling, and also influences the metrical analysis, since there is a preference for strong beats at changes of harmony (this is the reason for estimating the meter and harmony jointly). The stream segregation problem is solved with dynamic programming by grouping notes into streams such that the number of streams, the number and length of rests within streams, and pitch intervals within streams are all minimized [18].

The output of the probabilistic model is a list of beats, notes, and chord roots. Each beat includes an onset in milliseconds, and a level in a metrical hierarchy [17]. The probabilistic model considers the tactus and two subdivisions in the metrical structure; e.g., in a 3/4 meter, the tactus will be the quarter note, the first subdivision will be the 8th note, and the lowest subdivision the 16th note. The metrical structure also indicates the downbeats. Each note has an onset and a duration in milliseconds, a midi note number, and a stream number. The chord roots are quantized to the beats. An example of the output of this stage is shown in Fig. 4.

3.3 Quantize notes

The third step quantizes the note onsets to the closest beat subdivision. The offset of each note is also set to coincide with the onset of the next note in a stream; i.e., gaps within each stream are discarded. This avoids extra rests in the final scores, which could stem from notes played

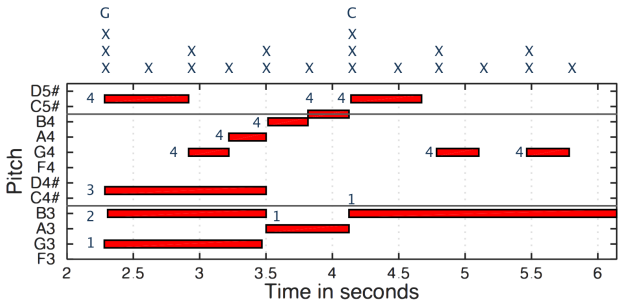


Figure 4. Sample output of the probabilistic model for estimating the metrical, harmonic, and stream structures. The Xs above the pianoroll illustrate the meter analysis (only 3 levels displayed). The letters above show the chord root (only roots on the downbeats are shown). The numbers next to the notes indicate the stream.

shorter than they should be. See, for instance, the two quarter notes in stream 4 in the second bar of the pianoroll in Fig. 4; they were played slightly shorter than 8th notes.

3.4 Determine note spelling

The correct note spelling is determined from the harmony generated by the probabilistic model and is based on the proximity in the line of fifths (the circle of fifths stretched out into a line) to the chord root. For example, the MIDI note 66 (F♯/G♭) would be spelled F♯ on a root of D, but spelled as G♭ on a root of E♭.

3.5 Assign streams to staves

The staves of the final score are set to be notated in treble clef for the upper staff and bass clef for the lower staff. Streams are assigned to the staff that accommodates all the notes with the fewest number of ledger lines.

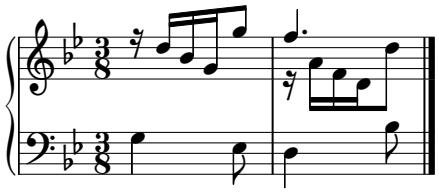


Figure 5. First two measures of Bach's Sinfonia in G minor, BWV 797. In the second bar, two streams are assigned to the same staff, so two separate monophonic voices must be created for proper rendering.

3.6 Detect concurrent voices

Once streams have been assigned to staves, we determine bars and voices. Bars are easily determined by the metrical structure, but note adjustments might be necessary if a note starts in one bar and continues to the next bar. In that case, the note has to be split into two or more tied notes. Concurrent notes in the same bar and staff must be detected and encoded appropriately for the next step. If a staff contains streams that overlap in time, we create monophonic voices consisting of sequences of notes. A sequence is defined as a gapless succession of notes and rests without overlaps. For example, as shown in Fig. 5, concurrent streams in measure 1 can be treated as monophonic inputs as they are assigned to separate staves, but in measure 2, two concurrent streams are assigned to the same staff, so we have to create two monophonic sequences of notes as input for the next step, one containing the F dotted quarter, the other containing the 16th notes and the D 8th note.

3.7 Generate the score

Finally, a Lilypond input file is generated. Lilypond is a free, command-line oriented music engraving program, which takes a text file as input and, thus, is suitable for the automatic generation of music notation. A possible alternative to Lilypond, which was considered during our research, is MusicXML [10]. Lilypond has the advantage of a simpler and more concise syntax. For instance, the music example from [10], which requires 130 lines of MusicXML, only requires 12 lines in Lilypond.

4. EVALUATION

To evaluate the proposed method, we asked five doctoral students in the Music Theory department of the Eastman School of Music, at various stages of advancement in their program, to blindly rate the output of the proposed method, two commercial programs (Finale 2015 [3] and GarageBand 10 [4]) and a free engraving program (MuseScore 2) applied to the Kostka-Payne dataset used to evaluate the probabilistic model [18]. The commercial programs have been chosen due to their popularity: GarageBand is freely available to all Mac users, Finale is one of the two major commercial music notation programs, the other being Sibelius. We also tested the import functionality of Sibelius but the results were very similar to the ones obtained by Finale, so we dropped this dataset to save time during the human evaluation. The dataset comprises 19 music excerpts, all of them piano pieces by well-known composers, for a total of 76 music scores to evaluate. The pieces were performed on a MIDI keyboard by a semi-professional piano player. For each piece we provided the original score, i.e., the ground truth. All the scores had been anonymized, so that the source program name was unknown, and the order of the evaluation was randomized. The evaluators were asked the following questions: 1) Rate the pitch notation with regard to the key signature and the spelling of notes. 2) Rate the rhythmic notation with regard to the time signature, bar lines, and rhythmic values. 3) Rate the notation with regard to stems, voicing, and placement of notes on staves. These three questions summarize the most important features that determine the formatting and the readability of a musical score. The three features are also fairly independent of each other.

The ratings were on a scale from 1 to 10 – 10 being the best. We instructed the evaluators to rate the scores to reflect how close each output was to the ground truth.

Finally, we told the evaluators that, since each rating may reflect multiple aspects of the notation, it was entirely up to their judgment to decide how to balance them (e.g., the relative importance of time signature, barline placement, and rhythmic values for the second question).

The results are shown in Figs. 6, 7 and 8. The ratings from each evaluator have been normalized (z-scores) by subtracting the mean and dividing by the standard deviation, and the results have been rescaled to the original range by setting their mean to 5 and their standard deviation to 2. The proposed method outperforms all the other methods in the first two ratings – pitch notation and rhythm notation – and ties for the top in median for the third rating – voicing and staff placement. Paired sign tests show that the ratings of the proposed method are significantly better than all the three baselines for the first two aspects, at a significance level of $p = 0.0001$. For the third aspect, the proposed method is superior to Finale and equivalent to MuseScore at a significance level of $p = 0.0001$, while the comparison with GarageBand is statistically inconclusive.

More work is needed in the note placement. One common aspect of music notation that has not been addressed in the proposed method is how to group concurrent notes into chords; we can see how that affects the output in Fig. 1 (d). In the downbeat of the first bar, the lowest three notes are not grouped into a chord, as in the ground truth (Fig. 1 (a)). This makes the notation less readable, and also introduces an unnecessary rest in the upper staff. A possible solution to this problem consists in grouping into chords notes that have the same onset and duration, and that are not too far apart, i.e., so that they could be played by one hand. A possible drawback of this approach is that it may group notes belonging to different voices.

Another limitation of the proposed method is the positioning of concurrent voices in polyphonic passages. Currently, the proposed method relies on the streams detected in step 2 to determine the order in which the voices are positioned in step 6. In polyphonic music, voices can cross so the relative positioning of voices might be appropriate for certain bars but not for others. A possible solution is to introduce another step between 6 and 7 to analyze each single measure and determine whether the relative positions of the voice is optimal or not. These two limitations affect the note positioning, reflected in the scores shown in Fig. 8. Finally, the probabilistic model does not always produce the correct results, especially with respect to beats and streams. A more sophisticated model may improve the rhythm notation and the note positioning, reflected in the scores shown in Fig. 7 and Fig. 8.

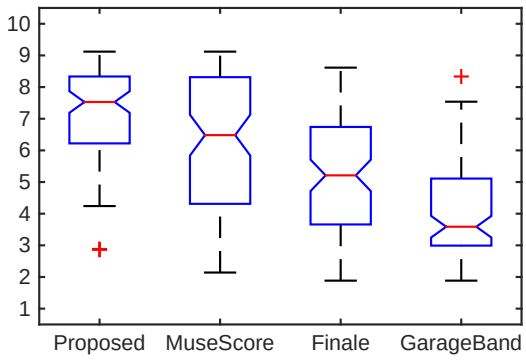


Figure 6. Normalized pitch notation ratings. Each box contains 76 scores from each of the 5 evaluators.

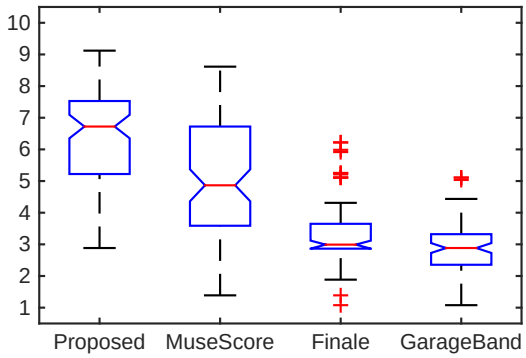


Figure 7. Normalized rhythm notation ratings. Each box contains 76 scores from each of the 5 evaluators.

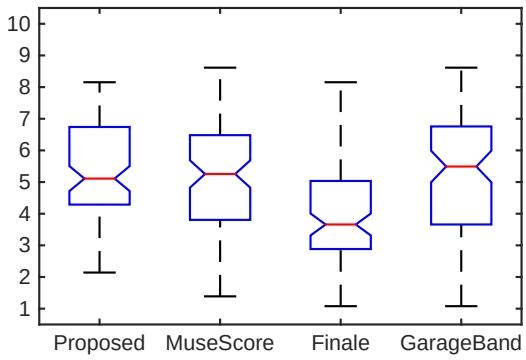


Figure 8. Normalized note positioning ratings. Each box contains 76 scores from each of the 5 evaluators.

5. CONCLUSIONS

In this paper we presented a novel method to generate a music notation output from a piano-roll input of a piano performance. We showed that the correct estimation of the meter, harmony and streams is fundamental in producing a properly formatted score. In a blind evaluation by professional music theorists, the proposed method consistently outperforms two commercial programs and an open source program in terms of pitch notation and rhythmic notation, and ties for the top in voicing and staff placement on 19 human performances on a MIDI keyboard. The proposed method can also be combined with any note-level automatic music transcription method to complete the audio to music notation conversion process, but more experiments are needed to assess the performance. For future work, we also plan to design a transcription metric for objective evaluation on a larger dataset, which should include complete piano pieces.

6. REFERENCES

[1] http://lilypond.org.

[2] https://musescore.org.

[3] http://www.finalemusic.com.

[4] http://www.apple.com/mac/garageband/.

[5] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[6] Emilios Cambouropoulos. From MIDI to traditional musical notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance and Analysis*, volume 30, 2000.

[7] Ali Taylan Cemgil. *Bayesian music transcription*. PhD thesis, Radboud University Nijmegen, 2004.

[8] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Piano music transcription with fast convolutional sparse coding. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6, Sept 2015.

[9] Tom Collins, Sebastian Böck, Florian Krebs, and Gerhard Widmer. Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[10] Michael Good. MusicXML for notation and analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001.

[11] Harald Grohganz, Michael Clausen, and Meinard Müller. Estimating musical time information from performed MIDI files. In *ISMIR*, pages 35–40, 2014.

[12] Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Proceedings 4th Sound and Music Computing Conference (SMC'2007)*, 2007.

[13] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[14] Kazuki Ochiai, Hirokazu Kameoka, and Shigeki Sagayama. Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 133–136. IEEE, 2012.

[15] Martin Piszczalski and Bernard A. Galler. Automatic music transcription. *Computer Music Journal*, 1(4):24–31, 1977.

[16] Haruto Takeda, Naoki Saito, Tomoshi Otsuki, Mitsuru Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. Hidden Markov model for automatic transcription of MIDI signals. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 428–431. IEEE, 2002.

[17] David Temperley. *Music and probability*. The MIT Press, 2007.

[18] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.