GRAME
COMPUTER MUSIC RESEARCH LAB.
Technical Report TR-120407

# Real-Time Score Notation from Raw MIDI Inputs

**D. Fober**
Grame - Centre national de création musicale
fober@grame.fr

**J. F. Kilian**
Kilian IT-Consulting
mail@jkilian.de

**F. Pachet**
Sony CSL
pachet@csl.sony.fr

## ABSTRACT

This paper describes tools designed and experiments conducted in the context of MIROR, a European project investigating adaptive systems for early childhood music education based on the paradigm of *reflexive interaction*. In MIROR, music notation is used as the trace of both the user and the system activity, produced from MIDI instruments. The task of displaying such raw MIDI inputs and outputs is difficult as no a priori information is known concerning the underlying tempo or metrical structure. We describe here a completely automatic processing chain from the raw MIDI input to a fully-fledge music notation. The low level music description is first converted in a score level description and then automatically rendered as a graphic score. The whole process is operating in real-time. The paper describes the various conversion steps and issues, including extensions to support score annotations. The process is validated using about 30,000 musical sequences gathered from MIROR experiments and made available for public use.

## 1. INTRODUCTION

Interactive Reflexive Musical Systems [IRMS] [1] emerged from experiments in novel forms of man-machine interactions, in which users essentially manipulate an "image" of themselves. Traditional approaches in man-machine interactions consist in designing algorithms and interfaces that help the user solve a given, predefined task. Departing from these approaches IRMS are designed without a specific task in mind, but rather as intelligent "mirrors". Interactions with the users are analyzed by IRMS to build progressively a model of this user in a given domain (such as musical performance). The output of an IRMS is a mimetic response to a user interaction. Target objects (e.g. melodies) are eventually created as a side-effect of this interaction, rather than as direct products of a co-design by the user.

This idea took the form of a concrete project dealing with musical improvisation, The Continuator. The Continuator is able to interactively learn and reproduce music of "the same style" as a human playing the keyboard, and it is perceived as a stylistic musical mirror: the musical phrases generated by the system are similar but different from those played by the users. It was the first system to propose a musical style learning algorithm in a purely interactive, real-time context [2]. In a typical session with the Continuator, a user freely plays musical phrases with a (MIDI) keyboard, and the system produces an immediate answer, increasingly close to its musical style (see Figure 1). As the session develops, a dialogue takes place between the user and the machine, in which the user tries to "teach" the machine his/her musical language.

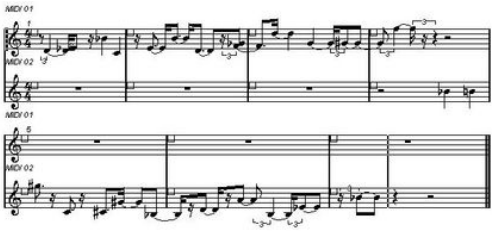

**Figure 1**. A simple melody (top staff) is continued by the Continuator in the same style.

Several experiments were conducted with professional musicians, notably Jazz improvisers [3] and composers such as György Kurtag. The idea to use the system in a pedagogical setting, with young children, came naturally. A series of exploratory experiments were then conducted to evaluate the impact and the new possibilities offered by the system in a pedagogical context. The results were more than promising, triggering a whole series of studies aiming at further understanding the nature of musical reflexive interaction [1]. All these experiments were based on a constraint-free system and extensions to integrate explicit pedagogical constraints are developed in the MIROR project framework.

In this context, the music notation is used as an analytic tool, reflecting both the children and the system activities. Although different, another score centered approach has been conducted in the VEMUS project [4], where the music score was used to convey feedback about students performance, using various annotations, including *objective* performance representations.

Converting the user and system performances into a music score is based on works taking place in the MIR field [5, 6] and in the music representation [7, 8] and rendering domain [9, 10]. The paper presents briefly the differ-

ent systems used for music representation, both at performance and notation level. Next it introduces the tools operating on these representations and shows how they collaborate. Annotating the music score at performance time is part of the features called by the analysis and that required to extend the low level music representation and conversion tools. These extensions are described by the section 5. The final section gives concrete uses and experiments conducted in the context of the MIROR project.

## 2. MUSIC REPRESENTATION

### 2.1 Score level

Score level music representation makes use of the GUIDO Music Notation format which has been presented in [9, 7, 8]. This paper will only recall the basic fundamentals of the format.

The GUIDO Music Notation [GMN] has been designed by H. Hoos and K. Hamel more than ten years ago. It is a general purpose formal language for representing score level music in a platform independent plain text and human readable way. It is based on a conceptually simple but powerful formalism: its design concentrates on general musical concepts (as opposed to graphical characteristics).

Notes are specified by their name `(a b c d e f g b)`, optional accidentals ('#' and '&' for sharp and flat), an optional octave number and an optional duration.

Tags are used to represent additional musical information, such as meter, clefs, keys, etc. A basic tag has one of the forms

```
\tagname
\tagname<param-list>
```

where `param-list` is a list of string or numerical arguments, separated by commas (','').

A tag may have a time range and be applied to a series of notes (e.g. slurs, ties, etc.); the corresponding forms are:

```
\tagname(note-series)
\tagname<param-list>(note-series)
```

In the following, we'll refer to *position* tags for the former and to *range* tags for the latter.

A GUIDO score is organized in note sequences delimited by brackets, that represent single-voices. Multi-voiced scores are described as a list of note sequences separated by commas as shown by the example below (Figure 2):

```
{ [ e g f ], [ a e a ] }
```

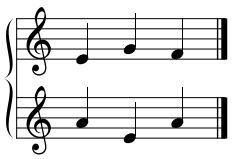

**Figure 2**. A multi-voices example

Below is an example of GUIDO notation, describing a four voices score with the corresponding output (Figure 3).

```
{
 [
  \barFormat<"system">
  \staff<1> \stemsUp \meter<"2/4">
  \intens<"p", dx=1hs,dy=-7hs>
  \beam(g2/32 e/16 c*3/32) c/8
  \beam(\noteFormat<dx=-0.9hs>(a1/16) c2 f)
  \beam(g/32 d/16 h1*3/32) d2/8
  \beam(h1/16 d2 g)],
 [\staff<1>\stemsDown g1/8 e
  f/16 \noteFormat<dx=0.8hs>(g) f a a/8 e
  f/16 g f e],
 [\staff<2> \meter<"2/4">
  \stemsUp a0 f h c1],
 [\staff<2> \stemsDown c0 d g {d, a}]
}
```

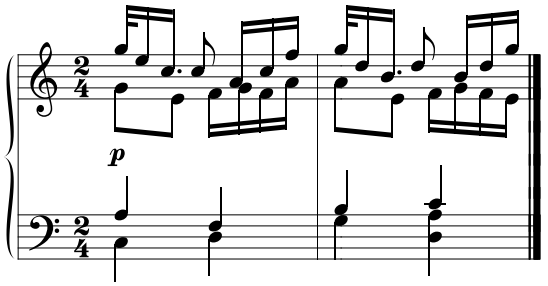

**Figure 3**. A four voices score.

### 2.2 Performance level

As representation format for the performance level input data The Continuator system uses the GUIDO-Low-Level-Notation (GLN) format.
As specified in [5] GLN has following features:

- text based file format.
- can be used as textual representation of any information expressed in a MIDI file.
- the syntax is compatible to GMN, except the usage of note and rest events is discouraged.
- it supports additional tags, not part of the GMN specification, e.g., `\noteOn`.

A simple set of rules is sufficient to convert a binary MIDI file into the equivalent textual GLN file:

- each track and all events of a single channel of the MIDI file get mapped to an individual sequence in the GLN file.
- any control change event of the MIDI file gets represented by the equivalent GLN tag.
- global control changes in MIDI track 0 get duplicated in every sequence of the GLN file.
- the integer based MIDI note names get converted into text based GMN pitch information and added as parameter of a `\noteOn` or `\noteOff` tag, e.g. MIDI pitch 60 gets mapped to c1.
- tick based timing information of the MIDI file gets converted into empty-note events with absolute timing (millisconds) in the GLN file.

Example for a GLN file:

```
{[
  \meter<"4/4">
  empty*5ms \noteOn<"g1",120>
  empty*440ms \noteOn<"a1",120>
  empty*5ms \noteOn<"c2",120>
  empty*4ms \noteOff<"g1",0>
  empty*240ms \noteOff<"a1",0>
  empty*4ms \noteOff<"c2",0>
  empty*1ms \noteOn<"f1",120>
  empty*230ms \noteOn<"d1",120>
  empty*4ms \noteOff<"f1",0>
  empty*1050ms \noteOff<"d1",0>
]}
```

The parameters of the `\noteOn` and `\noteOff` tags are pitch and intensity (MIDI semantic: 0...127).

## 3. FROM PERFORMANCE TO SCORE

On an abstract level the process of converting performance data into score information can be described as inferring an abstract representation from non-abstract data.
The transcriber has to distinguish between musical inaccuracies (musical noise) caused by player's perfection of interpretation (professionals) or technical imperfection (beginners).

### 3.1 Converting GLN to GMN

The here used algorithms for converting low-level symbolic musical data given as GLN string, to score level notation (in GMN format) are based on the approaches and implementation as described in [5].

The proceeded tasks for the conversion are divided into separated steps:

- pre-processing: creation of note events and preprocessing of timing information, similar to "noise reduction".
- voice separation and chord detection.
- ornament detection
- tempo detection
- detection of the time-signature (optional)
- quantisation
- inference of additional score level elements
- output as GMN

The following is meant to be a brief description of the algorithms for these steps as used in the described system. Refer to [5] for more details and a comparison with other existing approaches in this areas.

**Pre-processing** - Before starting with more advanced algorithms for transcribing the low-level data into score-level notation, associated `\noteOn` and `\noteOff` tags in the input data have to be detected and converted into a note entity with an onset time and a duration.
This module performs also a merge operation where small deviations between onsets and offsets of different notes will be eliminated. Small overlaps between notes will also be eliminated, primarily by cutting note durations.

The step can be described as noise reduction of the temporal order in the symbolic data. It can significantly increase the ouput quality of the following voice separation routine.

**Voice Separation** - The separation of polyphonic input data into voices representing sequences of (non-overlapping) notes and chords could actually be performed at any stage of the transcription process - especially before or after the tempo detection. As shown in [5] the later steps (e.g. tempo detection, quantisation) evaluate the context information of notes and depend therefore on the quality and correctness of this information.

The here used algorithm is capable of finding a range of voice separations that can be seen as reasonable solutions in the context of different types of score notation (e.g., only monophonic voices, only one voice including chords), multiple voices and chords. The type of output score (e.g. monophonic melody, two voice melody, piano score) can be controlled by a small number of intuitive input parameters for the algorithm. These parameters specify the costs (or penalty) that certain features of a possible voice leading (e.g. duration of rests between successive notes, ambitus of a chord, total number of a voice, size of an interval between successive notes). The optimum solution, i.e. the output with the minimum costs, is then calculated by a randomised local search algorithm.

In particular because the voice separation algorithm allows the creation of chords, the number of possible solutions increases exponentially. The usage of a simple, brute force algorithm for comparing all possible solutions for the voice leading instead of using an advanced optimisation algorithm would drastically reduce the performance of the implementation.

**Ornament Detection** - Before performing the remaining key steps for the transcription (tempo-detection and quantisation) ornamental notes get detected and filtered from the raw data. The two important effects of this step are:

- small ornamental notes will be hidden from the following steps.
- large groups of ornamental notes (e.g. trills) will appear as a single, longer note in the following steps.
- the ornaments can be rendered with their correct symbols, better readable for humans.

**Tempo detection** - The here implemented approach uses a combination of pattern matching (structure oriented) for inferring groupings of notes and statistical analysis for inferring score information for single notes. The approach works in two phases: first a pattern matching between patterns of a database - containing rhythmic patterns - is performed and then for all regions where no best matching pattern can be found a statistical tempo detection, evaluating only single notes, is performed. The pattern database is read from files in GMN syntax. It is therefore possible to use or provide patterns that depend

on the individual preference of the users.

**Time signature** - Because the pattern-based part of the quantisation approach is based on the time signature of the performance this module is located between tempo detection and quantisation. If the given input data already includes valid time signature information the execution of this module can be skipped.

**Quantisation** - The quantisation module is implemented as a context-based, multi-grid quantisation approach in combination with a pattern-based approach. Because the output of the system is intended to be a readable or at least displayable score, the execution of the quantisation module is mandatory. This ensures that the output file contains only rhythmical constructs which can be displayed in regular graphical scores. (For example, a score duration of 191/192 could not be displayed correctly in a graphical score.)

**Inference of score level elements** - Before creating the GMN output additional elements, like slurs or a key signature, can be inferred by optional modules in the implementation.

A key signature gets estimated by analysing the statistical distribution of the observed intervals. If the key signature is available a set of heuristic rules for correct pitch spelling can be applied.

Articulation related score elements - The implementation includes two rule-based modules for inferring slurs and staccato information. These modules are based on the comparison of the given performance data and the inferred score data and do not require a statistical analysis or specific algorithmical models.

**Output** - The final output module converts the internal data structure into files in GUIDO Music Notation syntax.

## 4. SCORE RENDERING

In the context of the MIROR project, the actual processing chain that goes from performance to the graphic score rendering is illustrated in Figure 4.

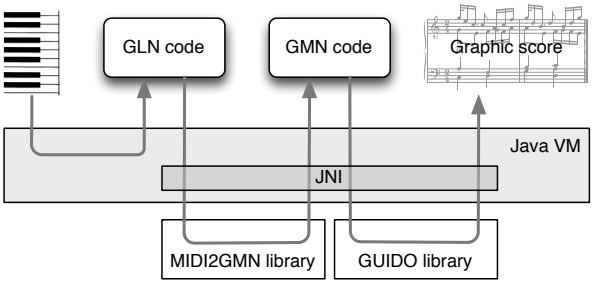


**Figure 4**. From performance to graphic score.

The system is implemented in Java but the key features - high level music representation inference and score rendering - are provided by C/C++ libraries through Java native interface [JNI]. Conversion from MIDI to GLN is achieved at Java level and is not described by this paper. Conversion from GLN to GMN is in charge of the MIDI2GMN library, an open source project [11] that implements the results of [5]. The MIDI2GMN library is implemented in ANSI C++ and can be compiled on any standard operating systems. Conversion from GMN to a graphic score is in charge of the GUIDO library, that is also an open source project [12], result of [10]. The project is also cross-platform and supports the main operating systems.

The whole processing is efficient enough to run in real-time i.e. to convert MIDI input to a graphic score while the user is playing.

## 5. PERFORMANCE ANNOTATION

It may be convenient to add annotations at performance time i.e. at GLN level, because the corresponding information is available at that time (e.g. to differentiate between user performance and generated continuations).

Since GLN and GMN share a common syntax, we can consider adding annotations using existing GMN tags (e.g. `\text<>`) interleaved with GLN code, without additional cost, at least at parsing level. However, this strategy is not straightforward, due to some fundamental differences between GLN and GMN to encode the music:

- GLN has no notion of note but a MIDI like semantic based on pairs of `\noteOn` and `\noteOff`
- GLN is organized as sequence of tags separated by empty-events [1] with no notion of chord or voice while chords and voices are explicit in GMN.

For most of the cases, GMN tags inserted in GLN code can be passed through the conversion process without further processing. Only a few cases need a special handling under the form of rewriting rules. In the remainder of this section, and when describing tags sequence, ***tag*** will refer to GMN position tags and ***rtag*** to GMN range tags (see section 2.1).

### 5.1 Tags inside a note

GLN describes note events in terms of `\noteOn` and `\noteOff` which allows tag insertion *inside* a note. While this shouldn't be the case for GLN files created directly from MIDI input, it could occur in the general case or particularly if the MIDI input gets processed by an advanced interactive system that adds these tags on purpose.
Since the first operation (see 3.1 Pre-processing) consists in grouping `\noteOn` `\noteOff` pairs in single note events, a decision has to be made for included tags.

Table 1 gives the rewriting rules for the tags included in notes: it consists in putting the tag outside the note.

When this rule is applied first, the remaining rules have only to deal with notes.

### 5.2 Tags inside a chord

Tags of the GLN input may be included into a chord in the GMN output. In many cases it doesn't make sense (e.g.

[1] An empty-event is a GMN event with a duration but no graphical representation in the score.

| GLN sequence | | GMN sequence |
|---|---|---|
| \noteOn ***tag*** \noteOff | ↦ | note ***tag*** |
| \noteOn ***rtag***(\noteOff) | ↦ | ***rtag***(note) |
| ***rtag***(\noteOn) \noteOff | ↦ | ***rtag***(note) |

**Table 1**. Rewriting rules for tags included in notes.

a meter or a key change). Table 2 gives the corresponding rewriting rules: it consists in putting the position tags outside the chord.

| GLN sequence | | GMN sequence |
|---|---|---|
| $n_1$ ***tag*** $n_2$ | ↦ | ***tag*** chord($n_1$, $n_2$) |
| $n_1$ ***rtag***($n_2$)) | ↦ | chord($n_1$, ***rtag***($n_2$)) |

**Table 2**. Rewriting rules for tags included in chords.

### 5.3 Tags and voice dispatched notes

GMN tags may be related to notes that are dispatched to different voices, i.e. to different note sequences. It raises syntactic issues since a range tag can't cover different sequences. Table 3 gives the rewriting rules for tags placed between voices: position tags remain attached to the next note and range tags are repeated over voices.

| GLN sequence | | GMN sequence |
|---|---|---|
| ***tag***$_1$ $n_{v1}$ ***tag***$_2$ $n_{v2}$ | ↦ | [***tag***$_1$ $n_{v1}$], [***tag***$_2$ $n_{v2}$] |
| ***rtag***($n_{v1}$ $n_{v2}$)) | ↦ | [***rtag***($n_{v1}$)], [***rtag***($n_{v2}$)] |

**Table 3**. Rewriting rules for tags included in voice dispatched notes.

### 5.4 Control sequence

A control sequence is a special sequence which content is duplicated on output, at the beginning of each voice. The control sequence must be the first sequence of the the GLN description. It should not contain any `noteOn noteOff` tags. It is typically intended to specify information like meter, key, clef, etc.

### 5.5 Example

Below is the GLN code of example 2.2 with additional annotations:

```
{[
  \title<"Annotations",fsize=16pt>
  \meter<"4/4">
   empty*5ms \noteOn<"g1",120>
   empty*440ms \text<"A">(\noteOn<"a1",120>)
   empty*5ms \noteOn<"c2",120>
   empty*4ms \noteOff<"g1",0>
   empty*240ms \noteOff<"a1",0>
   empty*4ms \noteOff<"c2",0>
  \noteFormat<color="red">(
   empty*1ms \noteOn<"f1",120>
   empty*230ms \noteOn<"d1",120>
   empty*4ms \noteOff<"f1",0> )
   empty*1050ms \noteOff<"d1",0>
]}
```

The conversion to GMN gives the following (also illustrated in Figure 5):

```
{[
  \title<"Annotations",fsize=16pt>
  \tempo<"[1/4] =121","1/4=121"> \meter<"4/4">
  \i<"ff",0.94> g1/4
  \text<"A",dy=17hs>({c2/8, a1/8  })
  \noteFormat<color="red">( f1/8  d1/2)
]}
```

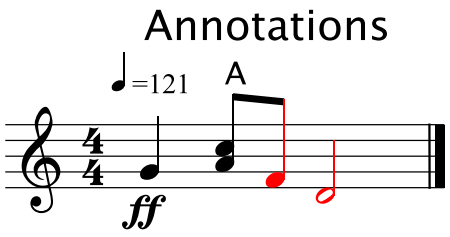


**Figure 5**. Annotated GLN to GMN conversion result.

## 6. PEDAGOGIC EXPERIMENTS

The processing chain presented in this paper was integrated in the MIROR-IMPRO and MIROR-COMPO software developed in the MIROR project. These software are designed to assist young children in improvisation and composition tasks, using the paradigm of reflexive interaction [13].

The goal of the score display is two-fold. Firstly, score display, especially when used in real-time, can be an interesting tool to sustain attention and encourage focused listening and playing behavior in children. Experiments were conducted to evaluate precisely the impact of visualisation on children playing styles and will be published shortly. Secondly, score display is used a posteriori by teachers to analyse the evolution of the musical skills of children during the year.

Technically, more than 30,000 musical sequences played by children or generated by the system have been successfully rendered. Figure 6 and 7 show examples of sequences typically played by children, both in piano-roll and score format. It can be observed that these sequences, played by unskilled people, are particularly complex to render. However, the score display provides a good approximation of the musical content that is musically more meaningful than the piano-roll.

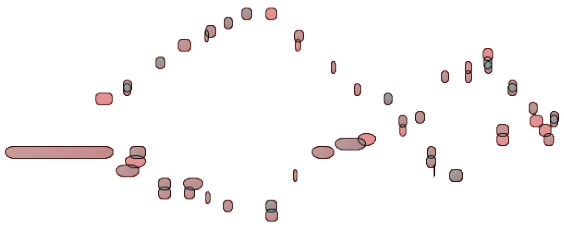

**Figure 6**. A typical sequence played by children in piano roll.

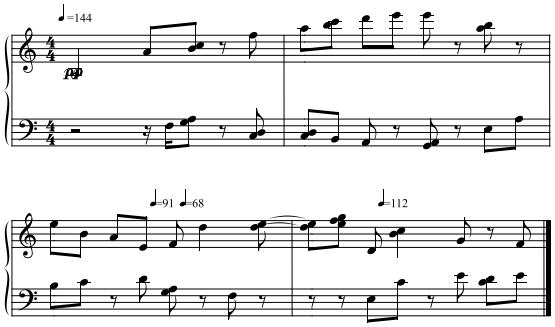


**Figure 7**. The score of a typical sequence played by children.

## 7. CONCLUSIONS

We described a processing chain to display high quality score representation from real-time MIDI input. This scheme was implemented and used by a Java software suite for children pedagogy. It proved robust and efficient enough for experiments involving intensive playing by unskilled users, which produce most difficult raw MIDI inputs to process.

There is still room for improvements, notably by optimizing the data flow path, i.e. converting MIDI to graphic score entirely at native level. This would require establishing a bridge between the midi2gmn and GUIDO Engine libraries.

The proposed extension for music annotation at performance level is designed to put together the best from the high level symbolic representation world with the immediacy of the real-time world.

All the components involved in the conversion and notation process are open source libraries available from SourceForge.

### Acknowledgments

This research has been partially supported by funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement #258338.

## 8. REFERENCES

[1] A.-R. Addessi and F. Pachet, "Experiments with a musical machine: Musical style replication in 3/5 year old children." *British Journal of Music Education*, vol. 22, no. 1, pp. 21–46, March 2005.

[2] F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[3] ——, "Playing with virtual musicians: the continuator in practice." *IEEE Multimedia*, vol. 9, no. 3, pp. 77–82, 2002.

[4] D. Fober, S. Letz, and Y. Orlarey, "Vemus - feedback and groupware technologies for music instrument learning," in *Proceedings of the 4th Sound and Music Computing Conference SMC'07 - Lefkada, Greece*, 2007, pp. 117–123.

[5] J. Kilian, "Inferring score level musical information from low-level musical data," Ph.D. dissertation, Technische Universität Darmstadt, 2004.

[6] J. Kilian and H. Hoos, "Voice separation: A local optimisation approach." in *Proceedings of the International Conference on Music Information Retrieval.*, 2002, pp. 39–46.

[7] H. Hoos, K. Hamel, K. Renz, and J. Kilian, "The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music." in *Proceedings of the International Computer Music Conference*. ICMA, 1998, pp. 451–454.

[8] H. Hoos and K. Hamel, "The GUIDO Music Notation Format Specification - version 1.0, part 1: Basic GUIDO." Technische Universität Darmstadt, Technical Report TI 20/97, 1997.

[9] D. Fober, S. Letz, and Y. Orlarey, "Open source tools for music representation and notation." in *Proceedings of the first Sound and Music Computing conference - SMC'04*. IRCAM, 2004, pp. 91–95.

[10] K. Renz, "Algorithms and data structures for a music notation system based on guido music notation," Ph.D. dissertation, Technische Universität Darmstadt, 2002.

[11] J. Kilian. (2012, Jan.) midi2gmn library. [Online]. Available: http://midi2gmn.sourceforge.net

[12] D. Fober. (2002, May) Guido engine library. [Online]. Available: http://guidolib.sourceforge.net

[13] F. Pachet, *The Future of Content is in Ourselves*. IOS Press, 2010, ch. 6, pp. 133–158.