MIR IN MATLAB: THE MIDI TOOLBOX

Tuomas Eerola
Department of Music
University of Jyväskylä, Finland

Petri Toiviainen
Department of Music
University of Jyväskylä, Finland

ABSTRACT (150-200 words)

The MIDI Toolbox is a compilation of functions for analyzing and visualizing MIDI files in the Matlab computing environment. In this article, the basic issues of the Toolbox are summarized and demonstrated with examples ranging from melodic contour, similarity, key-finding, meter-finding to segmentation. The Toolbox is based on symbolic musical data but signal processing methods are applied to cover such aspects of musical behaviour as geometric representations and short-term memory. Besides simple manipulation and filtering functions, the toolbox contains cognitively inspired analytic techniques that are suitable for context-dependent musical analysis, a prerequisite for many music information retrieval applications.

1. INTRODUCTION

MIDI Toolbox provides a set of Matlab functions, which provide versatile possibilities to analyze and visualize MIDI data. The development of the Toolbox has been part of ongoing research involved in topics relating to musical data-mining, modelling music perception and decomposing the data for and from perceptual experiments. The Toolbox is available free of charge under the GNU General Public License from http://www.jyu.fi/musica/miditoolbox/. Although MIDI data is not necessarily a good representation of music in general, it suffices for many research questions dealing with concepts such as melodic contour, tonality and pulse finding. These concepts are intriguing from the point of view of music perception and the chosen representation greatly affects the way these issues can be approached. MIDI format is also wide-spread among the research community as well as having a wider group of users amongst the music professionals, artists and amateur musicians.

The aim of MIDI Toolbox is to provide the core representation and functions for manipulating and analyzing MIDI files in Matlab. These basic tools are designed to be modular to allow easy further development and tailoring for specific analysis needs. Another aim is to facilitate efficient use and to lower the threshold for practical use. For example, the Toolbox can be used as teaching aid in music cognition courses.

2. GENERAL ISSUES

2.1 Representation

In the Toolbox, a MIDI file is represented as a matrix (hereafter *notematrix*) containing information about onset time, MIDI channel, pitch, velocity and duration of each note. Such notematrices can be created by a conversion function readmidi that reads in a MIDI file:

```
nm = readmidi('laksin.mid')

nm =
0  0.9   1  64  82  0     0.55
1  0.9   1  71  89  0.61  0.55
2  0.45  1  71  82  1.22  0.28
. . .
```

The columns of the matrix refer to (1) onset time in beats, (2) duration in beats, (3) MIDI channel, (4) pitch, (5) velocity, (6) onset time in seconds and (7) duration in seconds. The example notematrix given above are the first three notes of Finnish Folk song called *Läksin Minä Kesäyönä* (the notation is shown in Figure 1).

Large corpora of music can be conveniently processed in the Toolbox using a collection format, which stores multiple notematrices in a compact way and a set of meta functions that process the entire collection using a given analytical procedure or function. For example, the authors have created a collection of 8613 Finnish Folk Songs [1] using this format and also analyzed other music corpora using the same representation [2].

2.2. Functions

There are several categories of functions in the Toolbox that either manipulate, filter, analyze or generate notematrices. These functions are divided into following categories:

- Conversion functions
- Generation functions
- Filter functions
- Meta functions
- Plotting functions
- Statistical functions

- Key-finding functions
- Contour functions
- Segmentation functions
- Melodic functions
- Meter-related functions

Conversion functions read MIDI files into Matlab and export MIDI files from Matlab whereas *generation functions* allow creating and playing notematrices from within Matlab, either by invoking an external MIDI player or synthesizing the notematrix and sending the resulting waveform to a soundcard using Matlab's own `soundsc` function. Various *filter functions* allow editing the musical material, including scaling, shifting, quantizing, transposing and time-windowing notematrices. *Meta functions* are designed to apply any function for a collection of notematrices. *Plotting* and *statistical functions* are also helpful in visualizing the structure of a notematrix, either by displaying it using a pianoroll notation or showing the distribution of certain types of events of a notematrix. These functions facilitate use of the remainder of function categories which could be called analytic functions such as *key-finding, meter-finding, segmentation* and so on. Most of the functions in these categories involve cognitive models that are applied to a notematrix. Next examples of these analytical functions are demonstrated.

3. EXAMPLES

3.1. Melodic contour and similarity

Melodic contour describes the overall shape of the melody. The contour representation of a melody is usually easier to remember than exact interval information [3, 4] and numerous music informational retrieval systems make use of this information [5, 6]. Contour representation is available in the Toolbox using `melcontour` function. In Figure 1, two versions of melodic contour are plotted using two resolutions.

Melodic contour lends itself to different applications. For example, various distance measures can be used to calculate the similarity between the contour of melodic motifs or phrases. In the toolbox, distance calculations are handled by the `meldistance` function, which calculates the distance (or similarity) between two melodies using a user-defined representation (various distributions or melodic contour) and a distance measure. In this function, similarity can be scaled to range between 0 and 1, the latter indicating perfect similarity although this value does not indicate absolute similarity but is meaningful when compared to other melodic pair ratings.

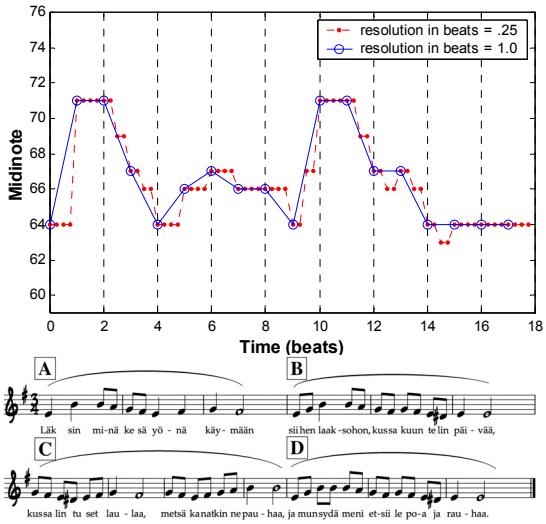

Figure 1. Melodic contour and the notation of an example tune, *Läksin Minä Kesäyönä*. Phrases are marked with capital letters. The contour plot depicts two versions of the melodic contour, sampled at a different resolution, for the first two phrases only (A and B).

For example, the similarity between the four phrases of the example tune (shown in Figure 1), using contour representation (20 samples) and city block distance (scaled between 0 and 1), is calculated as follows:

```
%% Load the melody from a selection of
%% reference tunes
nm=reftune('laksin_complete');

%% select the first 8 beats
ph{1} = onsetwindow(nm,0,8);

%% select the next phrases and trim
ph{2} = trim(onsetwindow(nm,9,17));
ph{3} = trim(onsetwindow(nm,18,28));
ph{4} = trim(onsetwindow(nm,29,37));

%% calculate the distances
for i=1:4
  for j=1:4
    dst(i,j)=meldistance(ph{i},ph{j},...
    'contour','taxi',20,1);
  end
end
```

	contour				**durdist1**		
Phrase	A	B	C	**Phrase**	A	B	C
A				A			
B	.90			B	.63		
C	.80	.76		C	.63	.95	
D	.90	1.00	.76	D	.54	.91	.89

Table 1. Melodic similarity of the four phrases of the example tune (notation shown in Figure 1) using different representations (rescaled between 0-1).

For the contour representation, phrases B and D are identical (similarity 1.00) and phrase C differs most

from the other phrases (Table 1). This seems intuitively reasonable although the exact numbers should be viewed with caution. However, similarity based on the distribution of note durations indicates greatest similarity between phrases B and C (.95) and lowest similarity between A and D (.54). The results of this simple indicator of rhythmic similarity are in contrast with the contour representation. These results are, again, readily apparent from the notated score.

3.2. Key-finding

The classic Krumhansl & Schmuckler key-finding algorithm [7], is based on key profiles obtained from empirical work by Krumhansl & Kessler [8]. In the K-S key-finding algorithm, the 24 individual key profiles, 12 major and 12 minor key profiles, are correlated with the pitch-class distribution of the piece weighted according to their duration. This gives a measure of the strength of each key. This method can be applied within a time window that runs across the length of the music to explore how tonality changes over time.

Let us take the Piano Sonata Nro. 1 in G major by F. J. Haydn (Hob. XVI:8). The first 12 measures of this sonata are shown in Figure 2. Below is an example of finding the maximal key correlation and obtaining the key at each two-beat interval:

```
%% Run K-S key-finding algorithm across
%% a 8-beat window moved by 4 beats
%% first calculate highest correlations
keys = movewindow(sonata,8,4,...
       'beat','maxkkcc');

%% then obtain the key names
labels = keyname(movewindow(sonata,...
        8,4,'beat','kkkey'));
```

The results of this are shown in panel B of the Figure 2. The plot displays the key changes over time, showing the local tonal center moving from G major towards D major. Although the measure shows the strength of the key correlation, it gives a rather simplistic view of the tonality as the dispersion of the key center between the alternate local keys is not shown. A recent dynamic model of tonality induction [9] calculates local tonality based on key profiles. The results may be projected onto a self-organizing map (SOM) trained with the 24 key profiles.

In the following example, the `keysomanim` function calculates the key strengths and creates the projection either as an animation in Matlab or as separate frames. The resulting maps underlying the tonal strengths are toroid in shape, which means that the opposite edges are attached to each other. This visualization of tonality can be used to show the fluctuations of the key center and key strength over time as an animation. Below is a static example of the `keysomanim` function using a window length of 4 beats and short-term memory time constant of 8 beats (the second parameter). The results are shown in panel C of Figure 2.

```
keysomanim(sonata,8,4,'beat','strip');
```

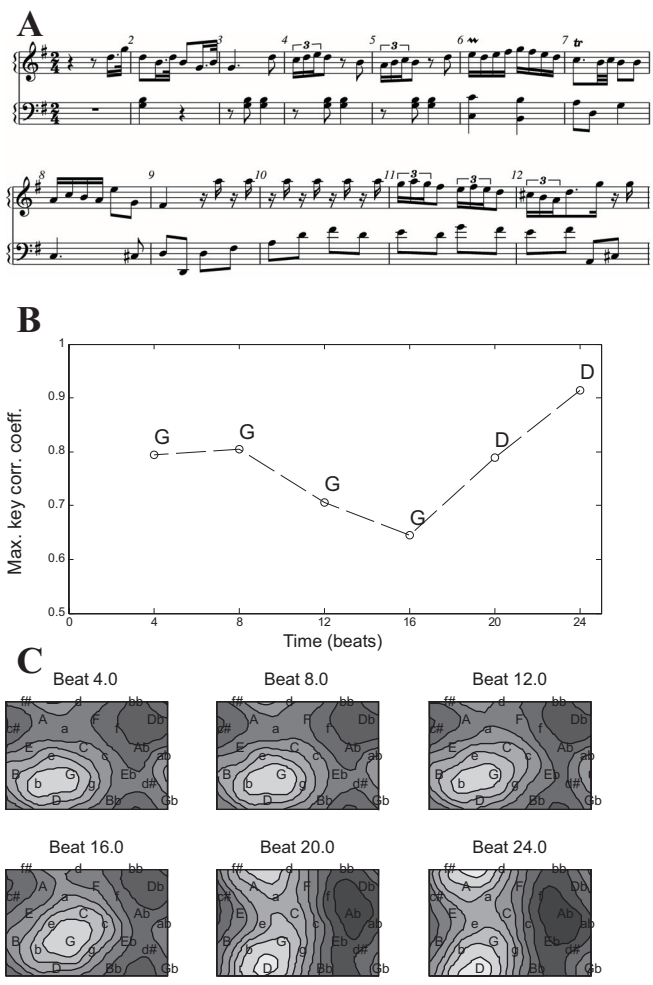

Figure 2. Panel A: Notation of the F. J. Haydn's Piano Sonata Nro. 1 in G major (Hob. XVI:8), first twelve measures. Panel B: Maximum key correlation coefficients across time (K-S algorithm [7] and the labels for most plausible local keys. Panel C: Dynamic model of tonality induction [9] portraying the local key centers across the excerpt.

From the separate figures of panel C one can see how the tonal center is first firmly in G major and then it slightly leans towards other regions, mainly D major. Another option in `keysomanim` function allows to save the animation as a Matlab movie ('movie'), which can also be written to a file using the `avifile` command.

3.3. Meter-finding

Inferring the meter is a challenge that involves finding a regular beat structure from a musical sequence. One technique is to use the autocorrelation function and to seek peaks from the onset structure corresponding to simple duple (2/4, 2/8, 2/2, 4/4) or simple triple meter (3/4, 3/2). This technique resembles the method used by Brown [10] to estimate meter. Toiviainen and Eerola [2]

tested the effectiveness of the method in classifying the meters into duple or triple using two large collections of melodies (Essen collection and Finnish Folk Tunes, N = 12368), [11, 1]. With only durational accents, the correct classification rate was around 80%. This method is available as the meter function in the Toolbox and applied to the Finnish folk tune *Läksin Minä Kesäyönä*:

```
meter(nm)
   ans = 3
```

This indicates the most probable meter is simple triple (probably 3/4). When melodic accent is incorporated into the inference of meter, the correct classification of meter is higher (up to 93% of the Essen collection and 95% of Finnish folk songs were correctly classified [2]). This optimized function is available in toolbox using the 'optimal' parameter in meter function, although the scope of that function is limited to monophonic melodies. Also, discriminating between compound meters (6/8, 9/8, 6/4) presents another challenge for meter-finding that will not be covered here. A plot of autocorrelation results – obtained by using `onsetacorr` function – provides a closer look of how the meter is inferred (Figure 3). In the function, the second parameter refers to divisions per quarter note.

```
onsetacorr(nm,4,'fig');
```

Figure 3 shows that the zero time lag receives perfect correlation as the onset distribution is correlated with itself. Time lags at 1-8 quarter notes are stronger than the time lags at other positions. Also, there is a difference between the correlations for the time lag 2, 3 and 4. The lag of 3 beats (marked with A) is higher (although only slightly) than the lags 2 and 4 beats and therefore it is plausible that the meter is simple triple.

Even if we now know the likely meter we cannot be sure the first event or events in the notematrix are not pick-up beats. In this dilemma, it is useful to look at the metrical hierarchy, which stems from the work by Lerdahl and Jackendoff [12]. They described the rhythmic structure of Western music as consisting of alteration of weak and strong beats, which are organized in a hierarchical manner. The positions in the highest level of this hierarchy correspond to the first beat of the measure and are assigned highest values, the second highest level to the middle of the measure and so on, depending on meter. It is possible to examine the metrical hierarchy of events in a notematrix by making use of the meter-finding algorithm and finding the best fit between cyclical permutations of the onset distribution and the Lerdahl and Jackendoff metrical hierarchies (Figure 4).

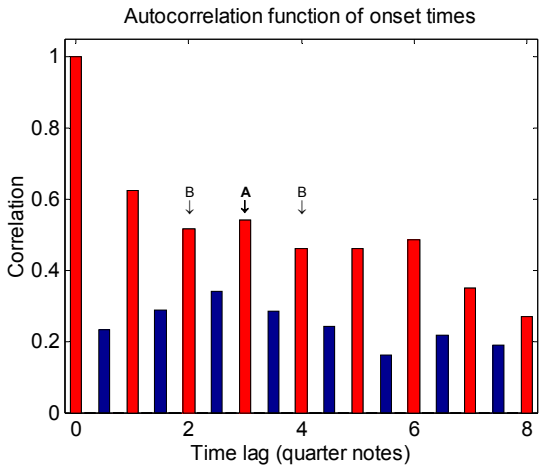

Figure 3. Autocorrelation function of onset times in *Läksin Minä Kesäyönä*.

```
plothierarchy(nm,'sec');
```

The dots in Figure 4 represent the metrical hierarchy. High stacks of dots (connected with a stem) correspond to events with high metrical hierarchy. In this melody, three levels are in use. The meter-finding algorithm infers the meter of the tune correctly (3/4), but the algorithm assumes that the first note is a pick-up note. This probably happens because of the metrical stress caused by the long notes in the second beats in measures three and six. A listener unfamiliar with the song could easily form this interpretation of meter.

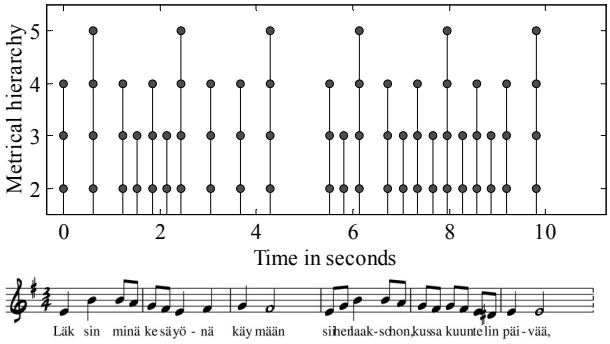

Figure 4. Notation and inferred metrical hierarchy of *Läksin Minä Kesäyönä*.

3.4 Melodic segmentation

One of the fundamental processes in perceiving music is the segmentation of the auditory stream into smaller units, melodic phrases, motifs and such issues. Various computational approaches to segmentation of monophonic material have been taken. With symbolic representations of music, we can distinguish rule-based and statistical (or memory-based) approaches. An example of the first category is the algorithm by Tenney and Polansky [13], which finds the locations where the changes in "clangs" occur. These clangs correspond to

large pitch intervals and large inter-onset-intervals (IOIs). This idea is partly based on Gestalt psychology. Another segmentation technique uses the probabilities derived from the analysis of melodies (e.g., [14]). In this technique, the probabilities of phrase boundaries have been derived from pitch-class-, interval- and duration distributions at the segment boundaries in the Essen folk song collection [11]. Finally, a Local Boundary Detection Model by Cambouropoulos [15] is a rule-based model that offers efficient segmentation. These segmentation algorithms are available in the MIDI Toolbox as individual functions. When applied to a simple folk song, *Läksin Minä Kesäyönä*, they produce segmentations shown in Figure 5.

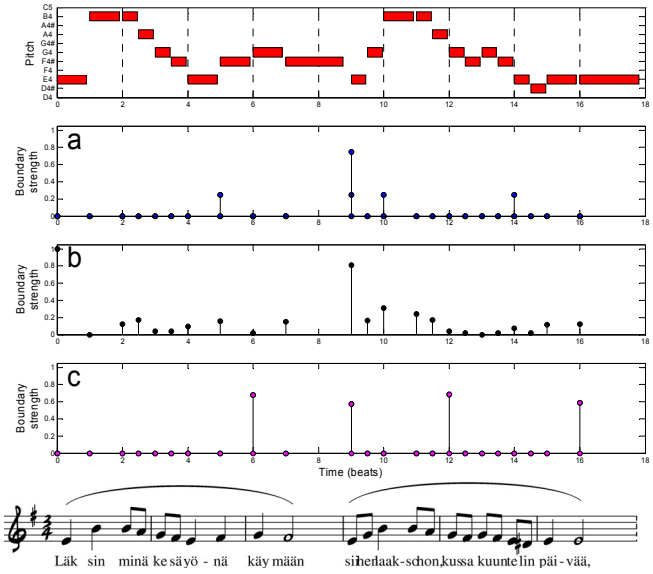

Figure 5. Three segmentations of *Läksin Minä Kesäyönä* showing the boundary strengths (0-1) based on (a) the Gestalt-based algorithm [13], (b) Local Boundary Detection Model [15] and (c) the probabilities of tone, interval, and duration distributions at segment boundaries in the Essen collection [11,14].

All segmentation algorithms produce plausible divisions of the example tune although the correct segmentation in the notation is most in line with Local Boundary Detection Model.

4. CONCLUSIONS

In this paper, the MIDI Toolbox has been shortly presented. The toolbox utilizes the efficient data processing and powerful visualization tools of Matlab. Currently the representation of MIDI files contains only note event information and hold pedal information. Future work will consider storing other information than the one pertaining to note events (e.g., key and time signature, copyright notes, track names, various types of controller data) in the notematrix. To make the Toolbox more convenient to use for novice users, a graphical user interface will be developed.

Though the Toolbox is based on symbolic musical data, the processing philosophy incorporates signal processing methods to cover such aspects of musical behaviour as short-term memory and geometric representations of contour. Compared with other systems available for manipulation of symbolic music (e.g., Humdrum [16], POCO [17], Melisma [18] and Rubato [19]), MIDI Toolbox connects directly with the powerful computational and visualization tools of Matlab. In addition, Matlab is available for the most operating systems and is widely used in the engineering community and hence it sports many readily available specialized toolboxes that are useful for music related operations (e.g., Statistics, Signal Processing, Neural Network and Fuzzy Logic Toolboxes).

Although many musically relevant tasks (such as recognizing a variant of a tune) may appear trivial for us as listeners, simulating them with a computer is challenging. We believe that cognitively inspired analytic techniques (segmentation, key-finding, meter-finding) are necessary to carry out this context-dependent task convincingly. We hope that the MIDI Toolbox will facilitate the application and development of such techniques within the MIR community.

5. REFERENCES

[1] Eerola, T., & Toiviainen, P. *Suomen kansan esävelmät: Digital archive of Finnish Folk songs* [computer database]. Jyväskylä: University of Jyväskylä, 2004. URL: http://www.jyu.fi/musica/sks/

[2] Toiviainen, P., & Eerola, T. "The role of accent periodicities in meter induction: A classification study". In S. D. Libscomb, R. Ashley, R. O. Gjerdingen, & P. Webster (Eds.) *Proceedings of the 8th International Conference on Music Perception & Cognition*, Evanston, IL (pp. 422-425). Adelaide, Australia: Causal Productions, 2004.

[3] Dowling, W. J. "Scale and contour: Two components of a theory of memory for melodies". *Psychological Review, 85(4),* 341-354, 1978.

[4] Dowling, W. J., & Fujitani, D. S. "Contour, interval, and pitch recognition in memory for melodies". *Journal of the Acoustical Society of America, 49*, 524-531, 1971.

[5] Kim, Y. E., Chai, W., Garcia, R., & Vercoe, B. "Analysis of a contour-based representation for melody". In *International Symposium on Music Information Retrieval*, Plymouth, MA: Indiana University, 2000.

[6] Lemström, K., Wiggins, G. A., & Meredith, D. "A three-layer approach for music retrieval in large databases". In *The Second Annual Symposium on*

Music Information Retrieval (pp. 13-14), Bloomington: Indiana University, 2001.
[7] Krumhansl, C. L. *Cognitive foundations of musical pitch*. New York: Oxford University Press, 1990.
[8] Krumhansl, C. L., & Kessler, E. J. "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys". *Psychological Review, 89*, 334-368, 1982.
[9] Toiviainen, P., & Krumhansl, C. L. "Measuring and modeling real-time responses to music: the dynamics of tonality induction". *Perception, 32(6),* 741-766, 2003.
[10] Brown, J. C. "Determination of meter of musical scores by autocorrelation". *Journal of Acoustical Society of America, 94(4),* 1953-1957, 1993.
[11] Schaffrath, H. *The Essen Folksong Collection in Kern Format.* [computer database] D. Huron (Ed.). Menlo Park, CA: Center for Computer Assisted Research in the Humanities, 1995.
[12] Lerdahl, F., & Jackendoff, R. *A generative theory of tonal music.* Cambridge: MIT Press, 1983.
[13] Tenney, J., & Polansky, L. "Temporal gestalt perception in music". *Journal of Music Theory, 24(2)*, 205-41, 1980.
[14] Bod, R. "Memory-based models of melodic analysis: challenging the gestalt principles". *Journal of New Music Research, 31*, 27-37, 2002.
[15] Cambouropoulos, E. "Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface". In M. Leman (Ed.), *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology* (pp. 277-293). Berlin: Springer Verlag, 1997.
[16] Honing, H. "POCO: An Environment for Analysing, Modifying, and Generating Expression in Music." In *Proceedings of the 1990 International Computer Music Conference* (pp. 364-368). San Francisco: Computer Music Association, 1990.
[17] Huron, D. *The Humdrum Toolkit: Reference Manual*. Menlo Park, CA: Center for Computer Assisted Research in the Humanities, 1995.
[18] Mazzola, G., & Zahorka. O. "The RUBATO Performance Workstation on NEXTSTEP." In *Proceedings of the ICMC* 1994. Århus, Denmark: ICMC, 1994.
[19] Temperley, D., & D. Sleator. *The Melisma Music Analyzer*. Available from http://www.link.cs.cmu.edu/music-analysis/, 2001.