A Sonification Approach to Music Visualization

Roberto Piassi Passos Bodo
University of São Paulo
rppbodo@ime.usp.br

Flávio Luiz Schiavoni
Federal University of São João del-Rei
fls@ufsj.edu.br

ABSTRACT

Music visualization always helped musicologists to analyze musical pieces. Traditionally, there are a few music visual formats that are standards and broadly used. Since computers started helping music analysis, several formats arose to represent music in a digital format. In this paper we propose three forms of music representation that can create visual feedback that is different from common music visualization. Our approach can bring some discussion about how different visual feedback can help musicians to understand a musical piece. This music representation is not concerned to be a better format but it is focused in aesthetics results that can provide alternative visualizations to musicians.

1. INTRODUCTION

Musicology uses the ears and the eyes to analyze music for centuries, and since the 1960's it is possible to use computers to aid this task. The first experience to automatically extract data from music, to analyze it with a computer, started with the researcher creating a music representation to use it as computer input, and using the computational power to compare data to analyze music [1]. In this work, it is presented a brief discussion about musical data representation with punch cards and the concern about how it could be easy to share musical data between research centers using a common music notation format. Since there were no music file formats in that age, they created a symbolic music notation to define music instructions that was called MIR (Music Information Retrieval) format [1]. Since then, it started a field to extract musical information, like style analysis, aided by computers [2].

Time passed and nowadays we have several different music file formats. From symbolic music (that stores events and notations) to audio files, passing by visual scores stored as images, and also text meta-data about music stored in catalogs. All these informations and file formats can be used to represent music in the computer and consequently can be used to extract informations about music [3]. Based on these formats, different tools to musicology arose using music representation and helping musicians on musical analysis, for instance JRing [4], Open Music [5], and Music 21 [6]. Also, there are huge databases available online with music files in different formats that can be used to extract information about music using statistics and machine learning techniques.

It is important to notice that different tools for music analysis can accept different file formats. The necessity of having different music representation happens because the kind of information that can be extracted from one format is different from the information extract from other format, and they can be complementary to help understand a composition or a musical piece. Some of them uses audio analysis while other uses symbolic music, for instance.

There are also the possibility to convert one music representation format into other format changing the music information point of view. Depending on how it is stored, it is feasible to edit, transpose, move, copy, and compare musical data. Also, it can be used to perform statistical analysis and plot graphics, and so on [6].

2. DATA SONIFICATION

A field that is not directly related to musicology but is also focused in transforming virtual musical content from one format to other is the so-called sonification. Classically, a goal of sonification is to transform complex multidimensional data into intuitive audio [7] as in text-to-speech or accessibility tools. Alternatively to this classic goal, it is possible to use sonification to create purely aesthetic sounds that can be used to inspire compositional process or just to be enjoyed by users [8].

Using this alternative approach of sonification, the web became a huge repository of data like text, images and web pages to be sonified and tried out as sonic elements (for instance, to compositional purposes or to performances). The process of sonifying data (such as HTML, image, and text) has some similarities with the process of synthesize symbolic music or even play audio files. Nonetheless, while traditional music notation formats, like MIDI, ABC or MusicXML has canonical sound mappings, other data types that does not have musical semantics need some aesthetic mappings to be played as music. Thus, using appropriated mappings it is possible to convert any data file to music information and listen to it.

Since the universe of mapping can be infinite and we cannot try every possibility to map web data information into music, some rational decisions can be taken to help this mapping process. A text can be sonified like ABC files or Lylipond format, where a character (or a group of) is mapped to a set of musical commands. The HTML file

format is similar to MusicXML since they are both hierarchical formats, and their structure can be mapped to music structure. Lastly, BMP images can be converted to WAVE files where the bytes of the audio samples are the bytes of the image pixels. A work presenting this sonification can be found in [9], including some sonification tools that create sounds from web pages, image files, and snippets of text, data that are not musical but that have similar structures with some digital musical formats.

3. FROM DATA SONIFICATION TO MUSIC VISUALIZATION

The set of mappings used in the sonification process has an interesting feature: they have well-defined inverses. These inverse mappings can be used to generate data files from music information as alternative representations for songs, such as a piano roll representing the list of MIDI events. One can convert two songs to text and look for the longest common subsequence between them to detect some melody similarity, or can convert one song to an image and detect repeated visual patterns in order to segment such music, and so on. Thus, these visual representations can be useful for music visualization, creating new possibilities and different approaches to music analysis.

What kind of web pages can we achieve using a MIDI score as HTML page structure? What kind of visual feedback can be reached using an audio file plotted as a bi-dimensional image? What kind of text we can get generating from a MusicXML score? These questions leaded the results of this research.

3.1 From score to HTML

When working with HTML sonification we did the mapping of seven CSS properties (width, height, top, left, padding, margin, border-width) to four synthesizer parameters (pitch, duration, dynamics, onset). The mapping functions were all linear: $synthProperty = a * cssProperty + b$. So, they have inverses in the following format: $cssProperty = c * synthProperty + d$, where $c = \frac{1}{a}, d = -\frac{b}{a}, a \neq 0$. Thus, also using linear equations we can convert music information to CSS properties to see different visual renditions of the same musical piece. Not only by varying the coefficients, but also by varying the mapped sound properties.

To this task we choose the MIDI file format as the input of this inverse mapping with the notes presented in Figure 1. MIDI protocol has the NOTE_ON event that contains the MIDI note number (pitch), the velocity (dynamics), and its own onset time. Lastly, with the respective NOTE_OFF event we can compute the duration of the note. With these data in hands we can convert the notes from a melodic sequence to elements in a HTML page. For instance, a simple conversion with duration to width, velocity to height, pitch to top, and onset to left, can generate a HTML page that is similar to a piano roll. In Figure 2 we have the x axis representing time and the y axis representing pitch (just like the piano roll), but we also have the height of the boxes showing different dynamics (the taller ones came from notes with higher velocities).

```
0, 0, Header, 1, 1, 120
1, 0, Start_track
1, 0, Control_c, 0, 7, 127
1, 0, Tempo, 600000
1, 0, Program_c, 0, 0
1, 0, Note_on_c, 0, 60, 120
1, 200, Note_off_c, 0, 60, 0
1, 200, Note_on_c, 0, 62, 110
1, 450, Note_off_c, 0, 62, 0
1, 450, Note_on_c, 0, 64, 100
1, 600, Note_off_c, 0, 64, 0
1, 600, Note_on_c, 0, 65, 90
1, 850, Note_off_c, 0, 65, 0
1, 850, Note_on_c, 0, 67, 80
1, 1000, Note_off_c, 0, 67, 0
1, 1000, Note_on_c, 0, 69, 70
1, 1250, Note_off_c, 0, 69, 0
1, 1250, Note_on_c, 0, 71, 60
1, 1400, Note_off_c, 0, 71, 0
1, 1400, Note_on_c, 0, 72, 50
1, 1650, Note_off_c, 0, 72, 0
1, 1650, End_track
0, 0, End_of_file
```

Figure 1. A C major scale presented as a CSV file.

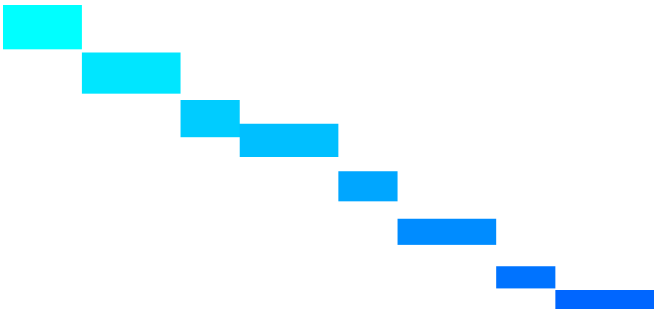

Figure 2. The visualization of a C major scale using the mapping: duration to width, velocity to height, pitch to top, and onset to left.

Other mappings can be explored, creating visualizations that are not so familiar (or traditional) as the piano roll, but that contain all the original information from the music piece, which is still easily recoverable. Figures 3, 4 and 5 show three alternative renderings of the same C major scale displayed at Figure 2 (the mapped properties are indicated in the respective captions).

The renderings displayed at the first three figures, we used width, height, top and left properties. Using only these four CSS rules, the musical attributes of the original song established the dimensions and positions of the HTML elements in a direct and explicit way. Observe that in Figure 3 higher pitches produced taller rectangles, longer durations made wider rectangles, bigger velocities placed the elements further down, and later onset times put

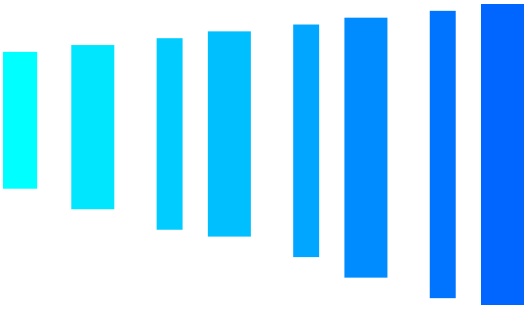

Figure 3. The visualization of a C major scale using the mapping: pitch to height, duration to width, velocity to top, and onset to left.

Figure 4. The visualization of a C major scale using the mapping: pitch to height, duration to top, velocity to left, and onset to width.

the elements farther to the right. In Figure 4 higher pitches produced taller rectangles also, longer durations put the elements further down, bigger velocities placed the elements farther to the right, and later onset times made wider rectangles.

One interesting feature of the chosen CSS properties is that we can use padding (internal gap) to control both dimensions of the HTML elements at the same time. In Figure 5 we can see that all displayed elements are squared, because we mapped duration to padding (observe that with longer durations, the squares became larger). Another feature is that we can use margin (external gap) to control together the vertical and horizontal displacement of the elements. Notice that in Figure 5 all HTML nodes are placed in a diagonal, because we mapped onset time to margin (then, with later onset times, the elements are further shifted to the lower right corner). Lastly, in Figure 5 once more, we mapped velocity to border-width, so we have thicker borders when the notes have faster attacks.

3.2 From audio to Image

When working with image sonification we did a data abstraction in which we used the image's pixels as the audio's samples. This was done using a flattened version of the image (which is normally represented by a matrix) to obtain an one-dimensional array of bytes for the audio. We only had to make some choices regarding if the image had 3 bytes per pixel (RGB images) or 1 byte per pixel (gray-scale images).

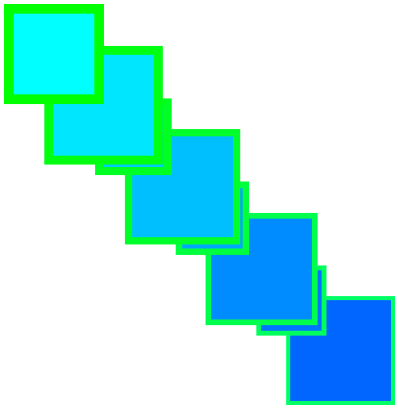

Figure 5. The visualization of a C major scale using the mapping: pitch to top, duration to padding, velocity to border-width, and onset to margin.

This abstraction works in the reverse order: we can use the audio's samples as the image's pixels. To use the bytes of the audio as the bytes of the image we need first to choose a value for image's width to be able to perform a reshape. Thus, we can convert a one-dimensional array to a 2D data structure. Again we have to be extra careful with the two basic types of images: RGB and gray-scale. Moreover, usually we have audio files with 16 bits per sample (Compact Disc standard), and this value does not match directly with 24-bit pixels, nor with 8-bit pixels.

For simplicity, we can consider an audio with 8 bits being transformed to a gray-scale image. The Figure **??** shows an image generated from an audio file containing a 210Hz sinusoid lasting 1 second (adding up 44,100 samples). The width of the image - 210 pixels - was chosen to put one complete period of the sinusoid per row of the image matrix. This choice aligned the pixels in a way that make the periodic pattern visually quite prominent.

Figure 6. A 8-bit wave file of a 210Hz sinusoid lasting 1 second in 44100Hz mapped to a image with 210 pixels of width.

A similar conversion has been made with an audio file containing a 441Hz sinusoid lasting 10 seconds (adding up 441,000 samples). The chosen width for the image was 1000 pixels (then the height is equal to 441 pixels). Thus, we get ten complete periods of the sinusoid per row. Again,

there is an alignment of the pixels making the periodic pattern easily seen (as we can verify in the Figure **??**).

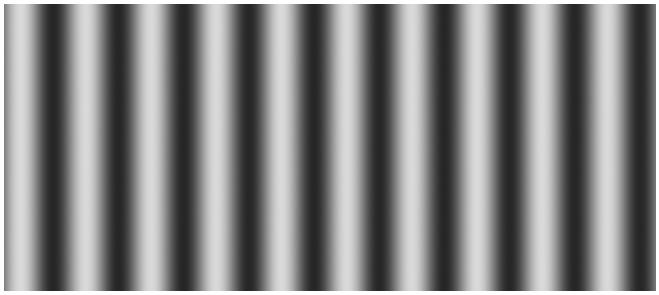

Figure 7. A 8-bit wave file of a 441Hz sinusoid lasting 10 seconds in 44100Hz mapped to a image with 1000 pixels of width.

Next, we tried musical pieces with richer spectrum. For instance, we concatenated two audios (a bass line with 7.5 seconds, and a guitar chord progression with 15 seconds) following the AABAAB pattern, and resulting in a 60 seconds audio (2,646,000 samples). Then, we convert it to a squared image ($width = height = 1627 \approx \sqrt{2,646,000}$) filling in the matrix with zeros when the bytes from the audio ended. In Figure 8 we can clearly see some visual patterns, and we can even recognize that they are repeated (we placed some white lines marking the start and the end of each segment).

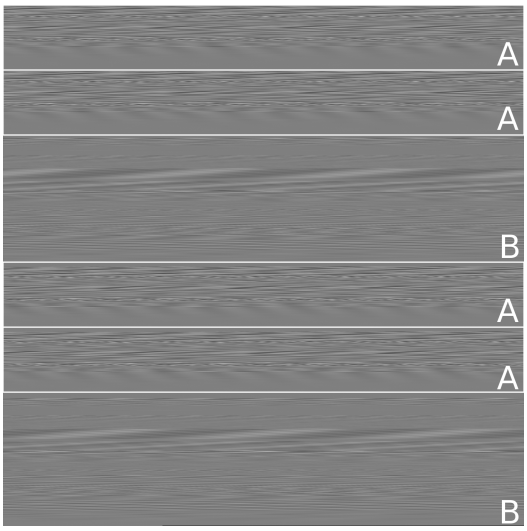

Figure 8. A 60 second audio file with subdivisions following the AABAAB pattern converted to a squared image.

In order to generate colored images, we did an alternative code that receives three 8-bit audios as input. It converts each audio file to a gray-scale image, and afterwards each image is assigned to a different color channel: the first image to Red channel, the second one to Green, and the third to Blue. Figure 9 shows four images: the first three ones are the gray-scale images generated from 882Hz, 441Hz, 220.5Hz sinusoids, and the last one is the RGB image generated from the three images above. Observe that the lighter colors in the RGB image are placed where the three gray-scale images are white, and the darker colors are placed where the three images above are black. We are also able to identify where each one of them is active alone (columns where the RGB image have red, green or blue colors), and even where they are active in pairs (columns where the RGB image have magenta, cyan or yellow colors).

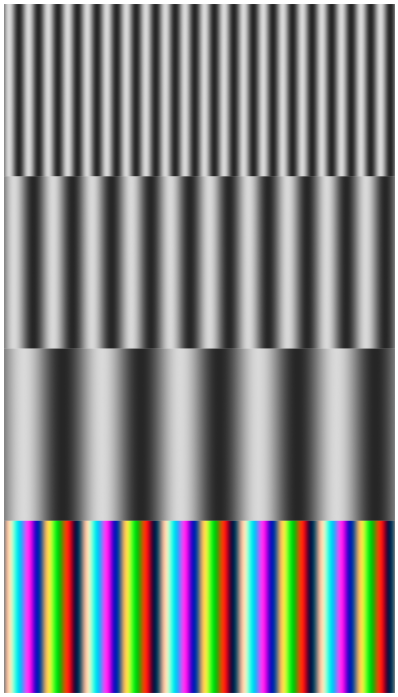

Figure 9. Three audio files being converted to a RGB image, each one being mapped to a different color channel.

These color combinations reflect the energy distributions in the original audio files. For instance, the points where the RGB image is white are the ones where the three audio files have energy peaks at the same time. The points where the pixels are black as the ones where the audios have energy valleys. Each other possible color can tell us which image was lighter at that point, thus it tells which audio had higher energy at the relative time.

3.3 From score to Text

When working with text sonification we implemented an algorithm that produces a melodic sequence mapping letters to notes. Because we were working with texts from web sites, we considered the most frequent letters in English to create the mappings. In this text to melody algorithm, we used the first twelve most frequency letters to determine the notes' pitches, the next eight ones to determine notes' durations (from 64th to double whole), one letter to produce rests and other letter to add a dot to a note [9].

In order to implement the inverse mappings for this sonifications process, we considered the MusicXML format, mainly because it has notes' durations by value. If we used MIDI, as in the audio to HTML mapping, we need to perform a quantization of the notes' durations in milliseconds to figure out their values, which is a delicate process prone to errors.

In Figure 10, we can see a typical note in MusicXML syntax, and three arrows pointing out the mappings: the

<note>
<pitch>
<step>C</step> ⟶ E
<octave>4</octave> ⟶ ,
</pitch>
<type>eighth</type> ⟶ m
...
</note>

Figure 10. Example of a single note from a MusicXML score.

p E , R , 0 , T , S , C , D , E -
p E - R - 0 - T - S - C - D - E .

Figure 11. A pitch shifting performed in a C major scale from the fourth octave (represented by the char ‘,’) to the fifth octave (represented by the char ‘-’).

C pitch is mapped to the ’E’ character; the 4th octave is mapped to ’,’ (whose ASCII value modulus 10 is 4); and the eighth-note is mapped to ’m’. A basic C major scale in the fourth octave - C4, D4, E4, F4, G4, A4, B4, C5 - with all durations as quarter-notes is mapped to the string: “pE,R,0,T,S,C,D,E-” (the “p” char appears once, because all notes have the same duration; if the last four notes were 8th-note the score would be “pE,R,0,T,mS,C,D,E-”).

This conversion from MusicXML to text has the potential to generate a symbolic sequence that can be easily compared. Computing the longest common subsequence (LCS), for instance, we can check the differences between two melodic lines. The LCS algorithm can show us variations such as additions, deletions and substitutions of characters from one sequence to the other. In the context of the text sequences that we generate from MusicXML files, this editions can mean a change in pitch, but also a change in the octave, or duration, or rest, etc.

In Figure 11 we have an example of a pitch shifting being performed in a C major scale from C4 to C5, changing it to a C major scale from C5 to C6. In Figure 12 we have an example of a time stretching being performed in a C major scale, changing all quarter-notes to eighth-notes. And in the Figure 13, we have samples for the three types of editions that we mentioned before: the deletion of a D4 note,

p E , R , 0 , T , S , C , D , E -
m E , R , 0 , T , S , C , D , E -

Figure 12. A time stretching performed in a C major scale from quarter-notes (represented by the char ‘p’) to eighth-notes (represented by the char ‘m’).

p E , R , 0 , T , S , C , D , E -
p E , 0 , T E , , S , C , T , E -

Figure 13. The deletion of a D4 note (represented by “R,”); the addition of a C4 note (represented by “E,”); the substitution of a B4 note (represented by “D,”) by a F4 note (represented by “T,”).

the addition of a C4 note, and the substitution of a B4 note by a F4 note.

4. DISCUSSION

The visualization of a score is a common way to extract data from music. Normally, visual music representation uses a common approach: x axis representing time and y axis representing pitch. This common approach is present in traditional scores, piano rolls and also in graphs plotted using symbolic music.

Our representations are probably out of this common approach and for this reason it can be weird, interesting, or useful, just because it explores different relations, representations, and mappings from the conventional approaches to display musical data.

We think these visualizations as part of a creative process, a combinational creativity [10] based on structural mapping from one parameter in sound to other parameters in visual content. Certainly, this is not a definitive form, a one-size-fits-all solution, and there are too much more to be experimented. The only certain we have is that our approach is a little bit out of formal methods to think visual music and we think it as an aesthetic form to visualize music.

The methods presented here are probably more suitable to bring some intuition about the music visually represented, or to open a discussion about other music formats, other ways to represent music and which music attribute is really intuitively represented in a graph. Furthermore, it can blur the frontier between web content and music content exposing common attributes from both sides.

If the images presented here can be useful to help someone to understand and analyze music, one can advocate that it can be possible to do the same work just listening to the music. In fact, it is, specially if you have good trained ears to to it. But if we intend to have a pervasive form to store the analysis, to teach it, to write about it, and to share it, it can be useful to think about music writing and visualization. Contemporary composers are always looking for other forms to write music than the traditional scores. Maybe we can inspire them to find new representations.

5. CONCLUSION

In 1857, the Frenchman Édouard-Léon Scott de Martinville invented the Phonautograph, the first device invented to record music. This equipment had an interesting feature: it records music but can not play it. The result of a record,

called phonautograms, could be used only to analyze music in a visual form.

With the advent of the computer, it was possible to create software to music analysis that are faster and more precise than that performed by human beings. Altogether, computer analysis do not replace human sensibility to understand and interpret music.

Nowadays, we have some different formats to represent and visualize music and every each format can present a kind of information that can be complementary to other representation and it is interesting to use more than one format to reach a better comprehension of a music piece.

Thinking about web content and starting from web sonification, we created a few mapping algorithms to music visualization based on common structured formats.

We considered all of the renderings very interesting and expressive. And they are just a glimpse of what the conversions, like MIDI to HTML code, are capable of doing. In this particular case, the actual coefficients of the linear mappings are chosen by the user, so this tool have an enormous potential that even the developers have not yet explored in full.

The three conversion tools that are presented in this paper are hosted at the following links:

- `https://github.com/rppbodo/midi2html`
- `https://github.com/rppbodo/musicxml2text`
- `https://github.com/rppbodo/audio2image`

6. ACKNOWLEDGMENT

We would like to thank the Computer Music Research Group at the University of São Paulo and the ALICE (Art Lab in Interfaces, Computers, and Everything else) from the Federal University of São João del-Rei. The author Roberto Piassi Passos Bodo would like to thank the support by SEMPRE: Society for Education, Music and Psychology Research.

7. REFERENCES

[1] M. Kassler, "Toward musical information retrieval," *Perspectives of New Music*, pp. 59–67, 1966.

[2] A. Mendel, "Some preliminary attempts at computer-assisted style analysis in music," *Computers and the Humanities*, vol. 4, no. 1, pp. 41–52, 1969.

[3] J. Futrelle and J. S. Downie, "Interdisciplinary communities and research issues in music information retrieval." in *ISMIR*, vol. 2, 2002, pp. 215–221.

[4] A. Kornstädt, "The jring system for computer-assisted musicological analysis." in *ISMIR*, Bloomington, IN, USA, 2001, pp. 93–98.

[5] J. Bresson, C. Agon, and G. Assayag, "Openmusic: Visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM International Conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 743–746. [Online]. Available: http://doi.acm.org/10.1145/2072298.2072434

[6] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," pp. 637–642, 2010.

[7] T. Hermann, A. Hunt, and J. G. Neuhoff, *The sonification handbook*. Logos Verlag Berlin, 2011.

[8] O. Ben-Tal and J. Berger, "Creative aspects of sonification," *Leonardo*, vol. 37, no. 3, pp. 229–233, 2004.

[9] R. P. P. Bodo and F. L. Schiavoni, "Synesthesia add-on: a tool for html sonification," in *Proceedings of the 16th Brazilian Symposium on Computer Music*, São Paulo - SP - Brazil, 2017, pp. 75–80.

[10] M. A. Boden, "Creativity and artificial intelligence," *Artificial Intelligence*, vol. 103, no. 1-2, pp. 347–356, 1998.