Measure-Based Scale-Aware Automatic Composition
by Marcelo Cicconet

1 Overview

We introduce an automatic music composition algorithm which has 3 main properties:
1. it is measure-based, what in this case means "chord-aware";
2. it is scale-aware: notes are chosen from a pre-defined musical scale;
3. except when the chord changes, consecutive notes are distant each other a fixed number of steps (on the chosen scale).

This algorithm was originally developed with de purpose of synthesizing a new song for a dance performance synthesized using a technique called *Motion Graphs* [2]. The overall process works as follows: first, a dancer performs listening to a song, while her movement is captured; then the captured movement is segmented and the segments are concatenated again (in a way different from the original sequence), observing some similarity between the borders of the segments.

It turns out that segmenting a movement is not an easy task. So, to simplify the process, the cuts were performed exactly in the points corresponding to the borders of the measures of the song, leading to what was called *measure-synchronous motion graph* [3]. Therefore, for the synthesized song to relate well with the synthesized movement, it should also be measure-synchronous.

The scale-aware aspects originated in the work [1], where the interest was in simulating the *target-note* improvisation approach, when the musician wants to hit a specific note in some point of the time. In that paper we used a Markov Process to generate notes, but the result did not sound good.

The third aspect is based in one of the most simple styles of improvisation: walking up or down a chosen musical scale, making steps of one, two or more notes. This is usually an early exercise that students practice when learning to play a musical instrument.

2 Some Music-Theory Aspects

In a music piece, the tonal properties are carried by the *melody* and *harmony*. Loosely speaking, they have to do with the sequence of notes and with the sequence of chords, respectively. As a general rule (at least for the *diatonic* scales) the chords are build from the notes of the chosen scale.

Let us pick the major diatonic scale, for instance. It has seven notes, and for each note we can build a *triad* chord (a set of three notes) whose *root* is that note. Specially when the chord changes, the first note after the change sounds nice if it is the root of the chord. That is what, in this text, we mean by "a note which sounds good with the chord".

3 Algorithm

We will describe the algorithm in pseudo-code, so it can more easily understood (and more easily implemented using the reader's preferred programming language). A Matlab implementation of the algorithm described here is available at `http://www.impa.br/~cicconet/autcomp`. The algorithm explains how to determine the notes for a particular bar.

$k \leftarrow$ "number of steps to walk up or down the scale"
$n \leftarrow$ "number of scale notes to be used in the composition"
$m \leftarrow$ "offset (left-most) note (in MIDI notation)"
$S \leftarrow$ "stencil of the scale (e.g., [2 2 1 2 2 2 1] for the diatonic major)"
$I \leftarrow$ "array of IDs, of size $|S|$, to be defined latter"
$N \leftarrow$ "array of notes, of size n, to be defined latter"
for $s = 1$ to $|S|$ **do**
  $I(s) \leftarrow s$
end for
$N(1) \leftarrow m$
$i \leftarrow 1$
for $j = 1$ to n **do**
  $I(j) \leftarrow i$
  if $j > 1$ **then**
    $N(j) \leftarrow N(j-1) + S(i)$
  end if
  $i \leftarrow i + 1$
  if $i = |S| + 1$ **then**
    $i \leftarrow 1$
  end if
end for
$n_0 \leftarrow$ "index of the note near which the composition will start"
$b \leftarrow$ "number of notes in the composition"
$i \leftarrow$ "the id of the chord used in the composition"
$M \leftarrow f(k, i, b, n_0, I, N)$

In the last statement, the function f returns the notes M for the bar. The pseudo-code for f is as follows.

$r \leftarrow$ "random number between 0 and 1"
if $r > 0.5$ **then**
$M(1) \leftarrow$ "first note above the note with index n_0 that has the ID i"
If there is no such note,
$M(1) \leftarrow$ "first note bellow the note with index n_0 that has the ID i"
else
$M(1) \leftarrow$ "first note bellow the note with index n_0 that has the ID i"
If there is no such note
$M(1) \leftarrow$ "first note above the note with index n_0 that has the ID i"
end if
for $j = 2$ to b **do**
$r \leftarrow$ "random number between 0 and 1"
if $r > 0.5$ **then**
$M(j) \leftarrow$ "note k steps above the note $M(j-1)$"
If there is no such note,
$M(j) \leftarrow$ "note k steps bellow the note $M(j-1)$"
else
$M(j) \leftarrow$ "note k steps bellow the note $M(j-1)$"
If there is no such note
$M(j) \leftarrow$ "note k steps above the note $M(j-1)$"
end if
end for

This is the basic algorithm, and of course many improvements can be made. Some are present in the Matlab implementation available at `http://www.impa.br/~cicconet/autcomp`.

References

[1] Marcelo Cicconet, Tertuliano Franco, and Paulo Cezar Carvalho. Plane tessellation with musical scale tiles and bidimensional automatic composition. In *International Computer Music Conference*, New York, NY, USA, 2010.

[2] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21:473–482, July 2002.

[3] Adriana Schulz, Marcelo Cicconet, and Luiz Velho. Motion scoring. In *ACM SIGGRAPH 2010 Posters*, SIGGRAPH '10, pages 13:1–13:1, New York, NY, USA, 2010. ACM.