2009

Flocking in the Time-Dissonance Plane

Adam James Wilson
CUNY New York City College of Technology

Recommended Citation

Wilson, Adam James. "Flocking in the Time-Dissonance Plane," in Proceedings of the 2009 International Computer Music Conference (ICMC 2009), Montréal, Canada, August 16-21, 2009, Michigan Publishing, University of Michigan Library, Ann Arbor, pp. 387-390. Online ISSN: 2223-3881. Permalink: http://hdl.handle.net/2027/spo.bbp2372.2009.087.

FLOCKING IN THE TIME/DISSONANCE PLANE

Adam James Wilson

Department of Music and Center for Research in Computing and the Arts
University of California, San Diego
ajwilson@ucsd.edu

ABSTRACT

This paper describes a technique for the sonification of an idealized model of the flocking behavior of birds, fish, and insects. Flocking agents are represented by pitches that move through time to produce chords of variable dissonance. The objective of each agent is to move toward more consonant chord formations with other agents. The output of the sonification is intended to provide material for use in musical composition.

1. INTRODUCTION

There are many musical adaptations of flocking behavior, made popular by Craig Reynolds' visual flocking simulation, *BOIDS* [4]. Perhaps the most facile musical analogies to be made to the group dynamics of birds, fish and insects involve some sort of surround-sound spatialization of sonic events. There are, however, many systems that use flocks and related phenomena to control other musical parameters. Blackwell's *Swarm Music* uses models of insect behavior to manipulate pitch, time, loudness, and duration, and to determine the frequency with which chords and sequences appear [1]. Unemi and Bisig implement a flocking "orchestra" in which three-dimensional positions of flocking agents are translated into pitch, panning, and velocity information [6]. My own implementation concerns itself with the perceived dissonance of collections of simultaneously occurring pitches; notes/agents are directed to "flock" toward consonant formations with one another.

In general, the flocking algorithm is used to describe the optimal movements of several independent agents as they attempt to maintain predetermined positions relative to one another and with respect to the position of a freely moving leader. Normally, the movements of the agents are confined to a plane. Each of the followers (1) observes the positions of the agents, including itself, (2) calculates the centroid of the agents' positions, (3) rotates and translates the predetermined pattern of desired positions around and along the line formed by the leader and the centroid, (4) observes the position of each follower relative to the available positions in the rotated and translated pattern, (5) selects the position in the projected pattern that is closer to its current position than to the positions of any of the other followers, and (6) moves incrementally to the selected position. Followers repeat these calculations and movements each time the leader changes position [2].

2. SONIFICATION OF FLOCKS

2.1. Mapping the Space

In the adaptation of the flocking algorithm described herein, the axes of the plane are defined by pitch and time. Pitch space is given by some musical interval broken into equal-tempered steps in the following manner:

$$s_i = \beta\lambda^{i/\phi}, \tag{1}$$

where $i = 0, 1, 2 \ldots (\phi - 1)$, β is the reference frequency, ϕ is the number of equal-tempered steps, and λ is the reference frequency coefficient. For example, assigning values $\beta = 440$, $\lambda = 2$, and $\phi = 72$ divides the octave between 440Hz and 880Hz into 72 equally separated frequencies.

Flocking agents are represented by notes, which have a location in time and a pitch. A pitch is a complex waveform that has a perceived fundamental frequency. The timbre of a given pitch is associated with the tone color of the sound source producing it, and has to do with the relative disposition of all sinusoidal components in the complex waveform. A chord is a collection of pitches, which is in turn a collection of partials, or sinusoidal components, each of which has a relatively stable frequency. Chords are formed by aggregates of flocking pitches; they are therefore reducible to collections of partials, whose relationships to one another, we will see, provide the basis for our moment-to-moment dissonance calculation.

Flocking agents begin by sounding the most dissonant chord possible given the timbral characteristics of the sound source, the resolution of the pitch space, and an invariant number of simultaneously presented pitches over time. Followers attempt to move into a more consonant relationship with each other and the leader. All notes are re-articulated simultaneously when either the leader or the followers move.

The dissonances of the possible sonorities are

determined prior to the computation of flocking behavior. The quality of the timbre used to articulate these sonorities is important; it must remain relatively invariant across pitch space. Dissonance values must be calculated for all sets of size n – the number of notes in each chord – for a particular timbre in all transpositions given by the equal-tempered divisions of the pitch space. For complex timbres and/or fine divisions of the pitch space, computation can be lengthy. It is therefore desirable to limit computation by including only the most significant sinusoidal components in the analysis of the timbre. This process is somewhat subjective; for the examples given in this description, I chose to exclude partials below -35dB, since very soft partials are unlikely to contribute significantly to the perception of dissonance.

It is also desirable to limit the pitch continuum to an interval of about an octave. Even for very complex timbres, intervallic dissonances become more and more subtle – eventually immeasurable – as notes move farther apart in frequency; at a certain point, the audible partials of the individual pitches no longer overlap. Conversely, very thick chords comprised of tones confined within an octave produce highly dissonant sounds that are technically distinguishable, but are perceptually quite similar. Chord size is therefore best limited to three or four pitches.

Once the set of chords is collected and the timbre to be used has been established, the perceived dissonance of the chords can be computed. Using the equations shown below, developed by William Sethares and incorporating an approximation of the Plomp and Levelt curves, we can calculate a perceptual dissonance value for two simultaneously presented sinusoids [5]. Since each available chord is a collection of sinusoids, we can sum the dissonance values of all unique combinations of two sinusoids in the chord to determine the dissonance of the chord itself. For all frequencies f_1 and f_2 and associated amplitudes a_1 and a_2, where $f_1 < f_2$, d (dissonance) is calculated as follows:

$$q = \frac{f_2 - f_1}{0.0207 f_1 + 18.96} \qquad (2)$$

$$d = a_1 a_2 [\, e^{-0.8424 q} - e^{-1.38 q} \,]$$

As chords become more consonant, their dissonance measurements approach zero. Two sine waves in unison, for example, will produce a dissonance value of zero. Different timbres moving in the same pitch space can produce very different dissonance curves over the same sets of pitches. Consider the two spectrograms shown in Figures 1 and 2. One represents an inharmonic gong with lots of spectral energy, and the other an electric keyboard with a simple harmonic spectrum, exhibiting few overtones. Compare the lists of consonant 3-note chord sets in Table 1. The values in the sets are 12-tone equal-tempered steps, where zero represents the reference frequency and eleven represents the step just below the octave above the reference frequency.

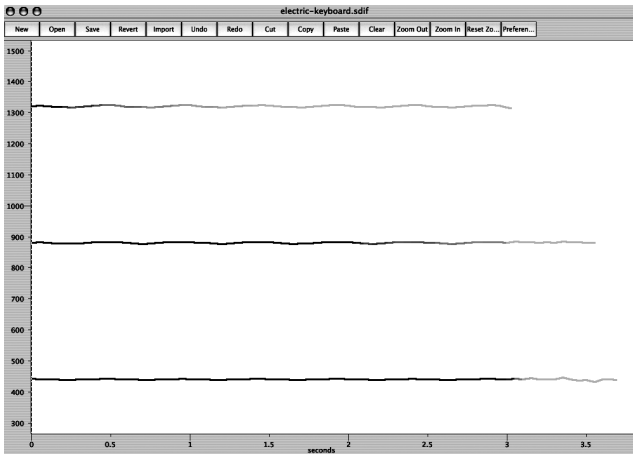

Figure 1. Electric keyboard partials (x-axis = time, *y*-axis = frequency).

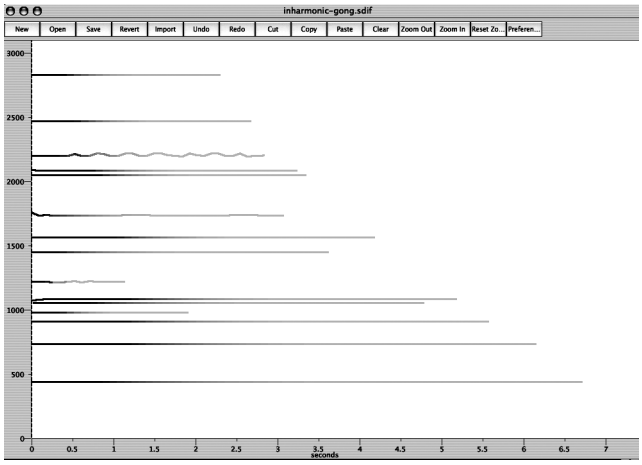

Figure 2. Inharmonic gong partials (x-axis = time, y-axis = frequency).

Most consonant electric keyboard chords.	Electric keyboard chord dissonance values.	Most consonant inharmonic gong chords.	Inharmonic gong chord dissonance values.
0 6 11	1.075×10^{-5}	0 8 11	1.491×10^{-4}
0 5 11	1.705×10^{-5}	0 3 11	1.567×10^{-4}
0 7 11	1.723×10^{-5}	3 8 11	1.571×10^{-4}
1 6 11	1.844×10^{-5}	0 9 11	1.600×10^{-4}
1 7 11	2.009×10^{-5}	3 6 11	1.603×10^{-4}
0 5 10	2.191×10^{-5}	2 7 10	1.608×10^{-4}

Table 1. Dissonance values for chords produced by two different timbres in the same pitch space. Numbers in columns 1 and 3 indicate 12-tet pitch classes.

2.2. The Algorithm

Followers step – literally moving incrementally by the smallest possible change in frequency – through a series of harmonies of undefined dissonance between the initial dissonant chord and the final consonance. The number of intermediate harmonies depends on the resolution of the pitch space, the number of equal-tempered steps between the furthest separated follower/target pitch pair, and whether or not a movement of the leader disrupts the progression of harmonies, immediately resolving the sonority to a consonant chord or provoking the followers to calculate new trajectories.

This model departs from typical applications of the flocking algorithm in two ways. First, there is no centroid calculation. Though the followers move into position through a two dimensional space, they move toward a common time-axis point, so calculations of relative "location" are made with respect to pitch only. Secondly, followers have the potential to move to one of *several* possible context-sensitive target configurations. The targeted pattern can change with each change in the absolute pitch of the leader. In this implementation, followers move to the most consonant chord containing the leader. The movement of the followers is cancelled if the leader leaps to a note that produces a greater consonance with the followers than the current target chord. The followers will not accidentally move to a chord that is more consonant than the target, because, again, the target chord *must* contain the leader, and the target chord is always defined as the *most* consonant chord containing the leader.

Once an acceptable consonance is reached, the algorithm resets, beginning with another dissonant chord, chosen this time at random from the five most dissonant chords in the list of possibilities. Although random leaps of the leader produce variations in progressions with identical initial and target chords, these variations aren't large enough to be aesthetically pleasing; the introduction of random choice for the starting chord produces wider variability across cycles of the algorithm.

2.3. The Algorithm, Step-by-Step

Here is a step-by-step description of the algorithmic process:

1. Analyze the timbre to be used; determine the most significant partials.
2. Establish divisions of the pitch space.
3. Establish the chord size.
4. Calculate the dissonances of all possible chords.
5. Specify the number of cycles to run. In all cycles except for the last, the leader will interrupt the movement of the followers with a leap.
6. Start the flock on the most dissonant chord, with the leader as the lowest-pitched voice.
7. Find the target chord – the most consonant chord that contains the leader.
8. For each follower, determine the note in the target chord that is closest in frequency to that follower; this is the follower's target pitch.
9. Move each follower stepwise toward its target.
10. If any follower is moving, even when others have reached their targets, re-articulate all notes.
11. If the next step of a follower will produce a unison with the leader, increase the step size by one for that follower for the next step only. Dissonances of chords containing unisons are undefined in this model.
12. If the leader leaps, check to see if the leap creates a chord that is more consonant than the target chord; if so, terminate movement and proceed to step ten.
13. If the target consonance is reached, proceed to step ten.
14. If there is more than one cycle left to go, start a new cycle by choosing a random chord from the five most dissonant chords, and begin again from step seven, with the leader starting as the lowest-pitched voice. Otherwise, go on to step eleven.
15. In the terminal cycle, the leader is prohibited from interrupting the progression toward the final consonance. Exit the process when this consonance is reached.
16. Render the movements of the flock as MIDI data.

2.4. An Example

Below is a sample call to the Common Lisp function "make-flock," which provides an interface to an implementation of the algorithm described herein. The results of the function call below are rendered in traditional music notation in Figure 3.

```
(make-flock
 :frequencies nil    ;; Supply a list of partials
                     ;; describing a timbre or
                     ;; specify a spear-infile
                     ;; below.
 :amplitudes nil     ;; Supply a list of amplitudes
                     ;; to go (in order) with the
                     ;; frequencies above, or
                     ;; specify a spear-infile below.
 :spear-infile "electric-keyboard.txt" ;; The path
                     ;; to a SPEAR[3] analysis file.
 :interval 2.1       ;; The upper-bound coefficient for
                     ;; transposing frequencies; the
                     ;; lower bound is 1.
 :divisions 72       ;; Equal tempered divisions of
                     ;; the transposition boundaries.
 :chord-size 3       ;; How many notes will the
                     ;; flocking harmonies contain?
 :combinations-infile nil ;; Skip calculation of
                          ;; chord combinations by
                     ;; opening a file containing a
                     ;; list of sets of indices
                     ;; corresponding to nCk, where n =
                     ;; divisions and k = chord-size.
                     ;; It doesn't make sense to
                     ;; specify this argument if
                     ;; chords-infile is non-nil,
                     ;; because in that case the
                     ;; calculation of combinations has
                     ;; already been taken care of.
```

```
:chords-infile "2.1-72-C-3_electric-keyboard.txt"
       ;; Supplying this argument causes the function
       ;; to skip calculation of note combinations
       ;; and subsequent ordering of chords by
       ;; dissonance.
:when-interrupt? .6  ;; How far through the motion
       ;; of the agent furthest from its target in a
       ;; given cycle should the flock be interrupted
       ;; by a leader jump, where, for interruption
       ;; time t, 0.0 <= t <= 1.0?
:cycles 4  ;; How many attempts should the flock make
       ;; to reach an optimal pattern, whether
       ;; thwarted or successful?
:duration 20  ;; How long (in seconds) should the flock
       ;; last?  The duration of each note is equal
       ;; to the total duration divided by the number
       ;; of chords in each flocking sequence minus
       ;; one (last note starts at the beginning of
       ;; the next "period").
:reference-frequency 92.499  ;; Sets the lower
       ;; bound for the pitch space; if not
       ;; specified, the lowest frequency partial
       ;; from the timbral analysis is used.

)
```

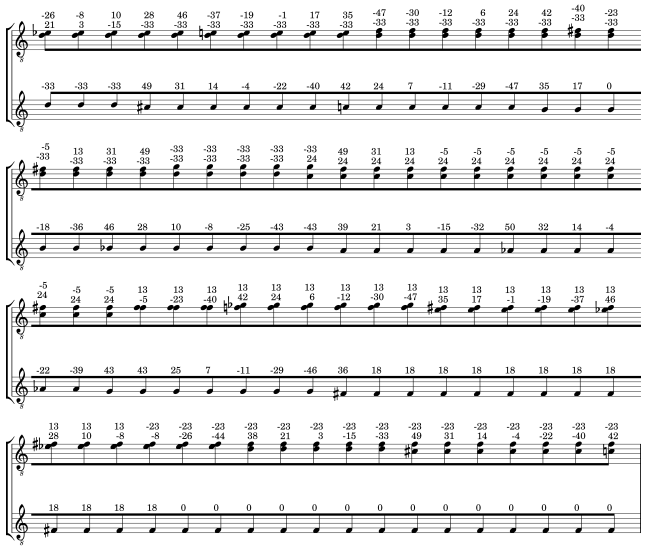

Figure 3. Flocking example. Numbers above notes indicate cents deviation from 12-tet pitches.

2.5. Potential Improvements

There are some possible modifications to this process that are worth consideration. For example, instead of moving to the most consonant chord containing the leader, which is also one of the most *distant* from the current chord in the spectrum of available dissonances, we can move to the *closest* chord containing the leader that is more consonant that the current chord. Similarly, we can target a consonance containing the leader that requires the least net movement of the followers. While these options are more in keeping with the spirit of the flocking algorithm, both will likely create chord progressions that seesaw rather quickly from dissonance to consonance. Furthermore, the difference in the perceived dissonance between the initial and target chords is likely to be quite small.

One aesthetic imperative here is to maximize the perception of motion from dissonance to consonance, but at the same time allow for harmonic variety. Currently, the chords that the followers step through are of undefined dissonance; only the target and the current chord are defined. This means that the perception of movement from consonance to dissonance may be distorted. It may be possible to rectify this by forcing notes to move to a consonant target chord only through chords of *intermediary* consonance, each containing the leader. Note that if this modification is *not* applied, it is only necessary to collect a small number of consonant chords prior to computation, beginning with the most consonant chord and moving incrementally through the spectrum of dissonances, until all of the possible absolute pitches are represented by the collection of chords.

There are also some possibilities for increasing harmonic variety. In the current implementation, the target chord for a given leader, expressed in absolute pitches, will always be the same. However, for different leaders, the target chord *quality* will not necessarily always be the same. Besides the technique – already employed – of starting each new cycle with a chord chosen at random from the five greatest dissonances, it may be useful to randomize which voice will be the leader, or to switch voices deterministically each time a new cycle begins.

Finally, it might be interesting to "apply some physics" to the model; for example, the followers currently all move at the same velocity – one discrete step at a time. Independently varying velocities would increase the rhythmic complexity of the results, but would have to be carefully tuned to avoid distorting the intended perception of harmonic relationships.

3. REFERENCES

[1] Blackwell, T. and M. Young. "Self-Organised Music," *Organized Sound*, vol. 9 no. 2, 2004, 137-150.

[2] Gervasi, V. and G. Prencipe. "Coordination without Communication: the Case of the Flocking Problem," *Discrete Applied Mathematics*, vol. 144, 2004, 324-344.

[3] Klingbeil, M. "Software for Spectral Analysis, Editing, and Synthesis," SPEAR Homepage, 2005. 25 Dec. 2007. <http://www.klingbeil.com/spear>.

[4] Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics, SIGGRAPH 87 Conference Proceedings*, vol. 21 no. 4, 1987, 25-34.

[5] Sethares, W. A. *Tuning, Timbre, Spectrum, Scale*. Springer-Verlag, London, U.K., 2005.

[6] Unemi, T. and Bisig. "Music by Interaction Among Two Flocking Species and Human," *Proceedings of the Third International Conference on Generative Systems in Electronic Arts*, 2005, 171-179.