Chapter 3

Baseline

3.1 K. Nymoen's bi-directional phase-adjustment

3.2 K. Nymoen's middle SA-leveled frequency-adjustment

3.3 System target state: harmonic synchrony

The state of harmonic synchrony is defined [1] as the state in which all agents in the musical collective "fire"/"flash", as described in Subsection 5.1.2, at an even and underlying interval or pulse, a certain number of times in a row. This is not to say all agents will have to "fire"/"flash" simultaneously, as has traditionally been the case for pulse-coupled oscillators []. Exactly how this can look is shown in Section 5.4, especially in Figure 5.1.

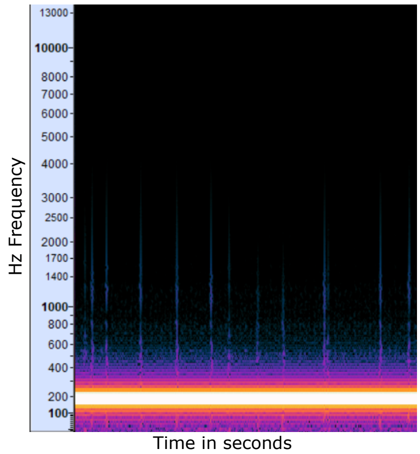
As one is designing and creating an interactive music technology system, one might want to encourage and allow for the playing of various musical instruments at various rhythms/paces, as it might be quite boring if all instruments were played at the exact same measure or pulse. As K. Nymoen et al. [1] reason when discussing their own interactive "Firefly" music-system, as well as coining the term of *harmonic synchrony*:

> *Temporal components in music tend to appear in an integer-ratio relation to each other (e.g., beats, measures, phrases, or quarter notes, 8ths, 16ths).*

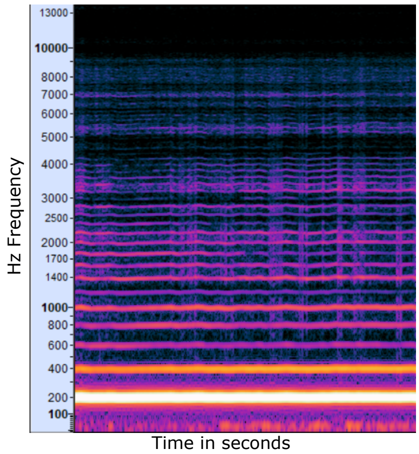
and

> *Being an interactive music system, people may want their device to synchronize with different subdivisions of a measure (e.g. some play quarter notes while others play 8ths).*

Accomodating for these aspects then, K. Nymoen et al. took inspiration for achieving synchronization in a decentralized system from the concept of *harmonics* in the frequency spectrum of a waveform, in that each harmonic wave or overtone has a frequency with an integer-relationship to the fundamental (smallest) frequency. This phenomenon can e.g. be seen in the frequency spectrogram of a humanly hummed G3-tone, depicted in Figure 3.1b, where one can observe the presence of harmonics and overtones having frequencies with integer relationships to the fundamental (smallest) frequency which was intended to be at 195,99 Hz.



(a) The frequency spectrogram of the audible waveform being a monotone and purely generated G3-tone at 195,99 Hz [2].



(b) The frequency spectrogram of the audible waveform being a more-or-less monotone but non-pure G3-tone, hummed and recorded by me [], as I tried to repeat the tone in 3.1a with my voice.

Figure 3.1: Frequency spectrograms of two different-sounding waveforms of the same G3-tone at 195,99 Hz. Note the absence and presence of harmonics and overtones in waveform 3.1a and 3.1b respectively. Frequencies in a harmonically synchronized agent collective will for the first ϕ-problem resemble the frequencies in 3.1a, where all frequencies are equal and constant. Conversely, when frequencies can be heterogenous and unequal, as in the ϕ- & ω-problem, the frequencies in a harmonically synchronized agent collective will rather resemble the frequencies in 3.1b, where these higher frequencies with integer-relationships to the fundamental and lowest frequency can be present.

More accurately then, and inspired by—although not completely analagous to integer-relationship frequencies like in Figure 3.1b e.g.—K. Nymoen et al. introduce the formal and "legal" requirement the oscillator-frequencies in the musical robot collective have to fulfill in order for the oscillator-frequencies to be harmonically synchronized. All musical agents—in a harmonically synchronized state—will have frequencies $\in \omega_0 \cdot 2^{\mathbb{N}_0}$, where ω_0 is the agent with the lowest frequency's frequency (i.e. the fundamental frequency), and $\mathbb{N}_0$ is the mathematical set of natural numbers including the number zero. Hence, agents will typically have frequencies like $\omega_0 \cdot 2^0 = \omega_0$, $\omega_0 \cdot 2^1 = 2\omega_0$, $\omega_0 \cdot 2^2 = 4\omega_0$, or $\omega_0 \cdot 2^3 = 8\omega_0$. If all agents end up with these kind of frequencies, we say they have "legal" and harmonically synchronized frequencies.

This state of *harmonic synchrony* is then the system goal state K. Nymoen et al. achieve using their phase- and frequency-update/-adjustment functions, as explained above in Section 3.1 and 3.2.

3.3.1 Detecting harmonic synchrony

In order to test and evaluate synchrony-performance in their firefly-inspired oscillator-system, K. Nymoen et al. [1] introduced some well-defined conditions the fireflies had to meet in order to be deemed *harmonically synchronized*:

- Firing may only happen within a short time-period t_f.
- Between each t_f, a period t_q without fire events must be equally long k times in a row.
- All nodes must have fired at least once during the evaluation period.

By utilizing transmitted firings/pulses from the robots in our robot-collective, these conditions can be enforced and checked throughout the synchronization-process, in order to detect if the oscillator-network becomes harmonically synchronized.

For getting a better idea of how these conditions being met looks like, see the *performance-measure plot* in Figure 5.1 where the oscillators/robots fulfill the abovementioned requirements right before ending the synchronization process.

These requirements, amongst other illustrations in Nymoen et al.'s paper [1], thus constitutes a blueprint for the design of a performance-/synchrony-measure able to detect the achievement of harmonic synchrony in a decentralized network of "firing"—or pulse-coupled—oscillators. The time having passed from the start of the synchronization-process until the detection of harmonic synchrony will then be defined as the performance-score, indicating how fast or slow the oscillators are at synchronizing.

The exact details of how such a performance-/synchrony-measure is implemented for our musical multi-robot oscillator-network, in the synchronization-simulator, will be given in Section 5.4.

Chapter 5

Implementation

5.1 Simulator setup: the musical multi-robot collective

5.2 Synchronizing oscillator-phases

5.3 Synchronizing oscillator-frequencies

5.4 Performance-measure: time until harmonic synchrony is detected

Our performance-measure will be used to evaluate and test our multi-robot collective's ability to harmonically synchronize to each other. As mentioned in Subsection 3.3.1, K. Nymoen et al.'s requirements for achieving *harmonic synchrony* serve as a blueprint or guide for how to implement our synchrony-/performance-measure. This performance-measure should be able to, during synchronization-simulation, detect if harmonic synchronization has been achieved in our decentralized oscillator-network. The successful triggering of this detection will then in turn terminate the synchronization simulation-run and save to a dataset the time it took to synchronize (the performance-score), in the case of a 'synchronization-success' — an example of which can be seen in Figure 5.1. If a certain amount of simulation-time has gone without the detection of harmonic synchrony occuring, the synchronization simulation-run is still terminated and datapoint still saved, but this time as a 'synchronization-fail'.

The resulting and corresponding performance-scores obtained using this performance-measure will then take values of the simulation-time (in seconds) it takes for the robot-collective, from the start of the synchronization-simulation, to achieve the system target state of *harmonic synchrony*, as specified in Section 3.3.

My specific implementation of the performance-/synchrony-measure essentially consists of enforcing all the requirements or rules listed in 3.3.1, given some constant t_f- and k-values (e.g. $80ms$ and 8 respectively [1]). And again—to recall from 3.3.1—t_f is the short time-window within which nodes are allowed to fire at each beat, and k represents how many times nodes have to fire at

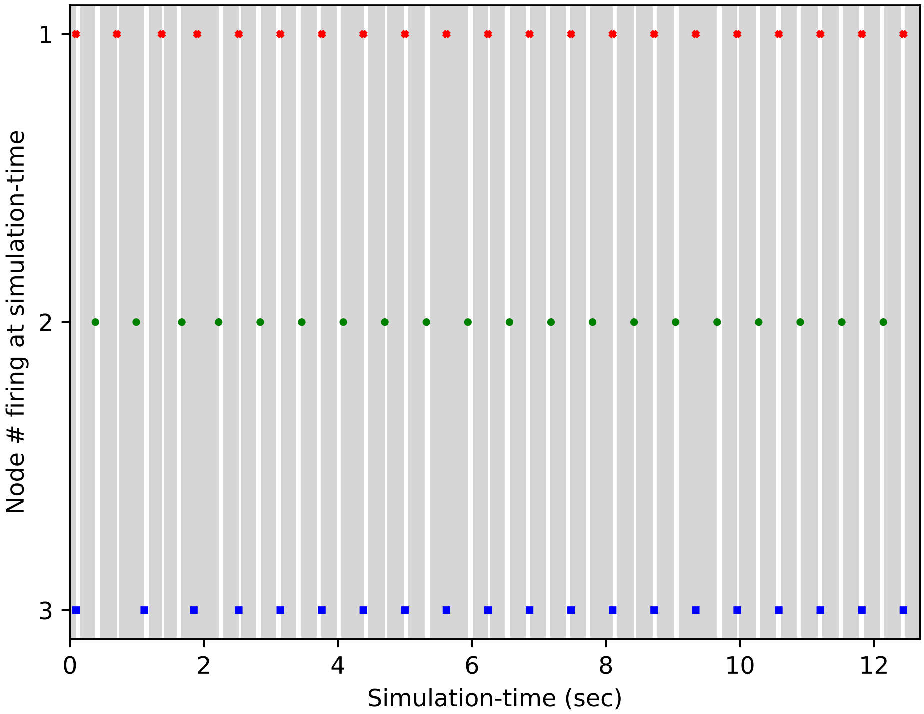


Figure 5.1: A **performance-measure plot**, displaying the temporally recorded pulses/“firings” transmitted by the robots in the Dr. Squiggles-collective throughout the synchronization-simulation in Unity. Short and white windows/strips in the figure represent the short ‘legal firing’ time-periods t_f during which nodes are allowed to fire within—unless the t_q-duration was just reset (in that case they are $t_f/2$ long, in order to in the future align pulses in the center of the t_f-windows). The larger gray windows represent the ‘silent’ time-periods within which no nodes are allowed to fire—if the agent-collective is to be harmonically synchronized. In this particular simulation-run above, the robots had to fire evenly $k = 8$ times in a row, within $t_f = 80ms$ long time-windows. As we can observe, harmonic synchrony was eventually achieved after around 12.5 seconds—thereby terminating the simulation-run in Unity as a success (and behind the scenes saving a datapoint consisting of the success-result as well as the 12.5 seconds performance-score, along with the simulator-hyperparameters, to a dataset).

even underlying pulses/beats in a row without changing the t_q-period—before becoming harmonically synchronized.

The requirement of firing evenly k times in a row with identical t_q-periods can be—and in fact is in our implementation—enforced by incrementing an integer variable *towards_k_counter* after a 'legal' t_f-window has occured (i.e. one or more nodes fired inbetween the onset and ending of the t_f-window), and conversely by resetting *towards_k_counter* to 0 when an illegally transmitted firing was heard during a 'silent' (or so it was supposed to be at least) t_q-window, hence restarting the synchrony-detection process—as can be seen in Figure 5.2.

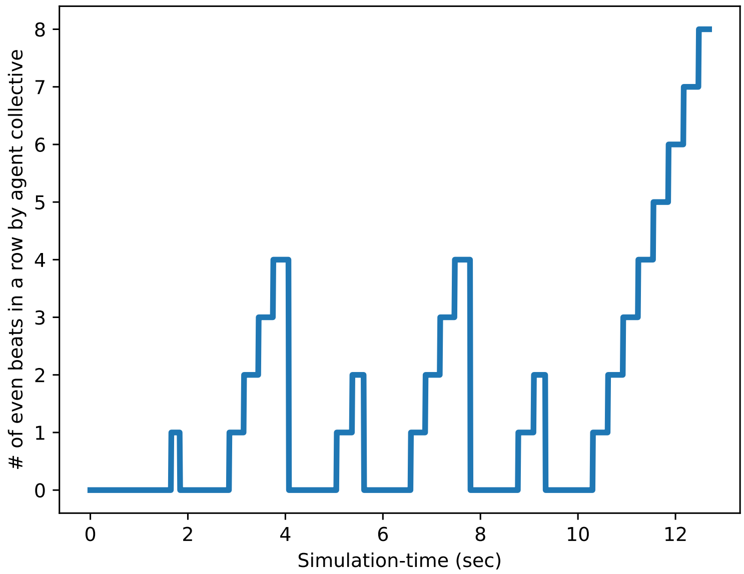


Figure 5.2: A **synchrony-evolution plot**, displaying the temporal recording of the *towards_k_counter*-variable throughout a synchrony simulation-run in Unity. The counter is incremented as the robot-collective fires evenly within 'legal' t_f-windows, and is conversely reset to 0 if illegal firings during 'silent' t_q-windows are heard. Note that in this specific simulation-run above (same run as in Fig. 5.1), the agents were on their way to achieve harmonic synchrony five times before the 10th second already, but since one or more of them fired 'illegally' (i.e. inside a t_q-window), they were consequently 'punished'—or rather deemed 'not synchronized enough yet'—by getting their counter reset to 0. Eventually however, through further phase- & frequency-synchronization, the multi-robot collective was after 12.5 seconds able to achieve harmonic synchrony, when *towards_k_counter* became equal to k, as well as all other requirements for achieving *harmonic synchrony* was met.

Initially, the t_q-period/-window is not initialized, as it entirely depends on the frequencies to which the robot-collective converges to; however, when an illegal firing (i.e. a firing perceived during a t_q-window) occurs—t_q is also then reset itself to a hopefully more correct value. (Regner kanskje med jeg bør utdype litt mer nøyere her med figur, matte, og evt. algoritme-pseudokode.. Eller hva?)

Bibliography

[1] Kristian Nymoen et al. "Decentralized harmonic synchronization in mobile music systems". In: *2014 IEEE 6th International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2014, pp. 1–6.

[2] Tomasz P. Szynalski. *Online Tone Generator*. URL: `https://www.szynalski.com/tone-generator/` (visited on 02/02/2022).