Computational Self-Awareness in Musical Robotic Systems

David Thorvaldsen

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Institute for Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

Computational Self-Awareness in Musical Robotic Systems

David Thorvaldsen

Computational Self-Awareness in Musical Robotic Systems

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

MSc Thesis/Project Summary.

Acknowledgements

Tusen takk til alle som har støttet og hjulpet meg. Sånn faktisk.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Grunnene til å studere effektene av selv-bevissthet.

De viktigste bidragene av MSc thesis-arbeidet summert (for å få det til å stå frem/ut bedre enn å bare "gjemme" det i den siste delen av oppgaven).

1.2 Goal of the thesis

BESKR.: ["to make the reader better understand what the thesis is about"—Jim, og "en rød tråd?"—Sigmund].

Spesifikke mål (goals/aims) med master-oppgaven/-prosjektet. Hva jeg vil vise/demonstrere til folk (leserne f.eks.) om self-awareness (SA) og hvordan.

Research Question 1:
Will performance in collective multi-robot systems increase as the level of Self-Awareness increases? Specifically, will increased levels of Self-Awareness in the individual agents/musical robots lead to the collective of individuals being able to synchronize to each other faster than with lower levels of Self-Awareness?

Research Question 2:
Will increased levels of Self-Awareness lead to more robustness and flexibility in terms of handling environmental noise and other uncertainties — specifically in the continued ability of musical robots to synchronize to each other efficiently despite these difficulties/challenges?

1.3 Outline

En fin (Eagle's-eye) oversikt over strukturen til hele dette dokumentet fra nå av, og utover.

Chapter 2

Background

Kulepunkter (bulletpoints) fra mulige inspirasjoner og referanser (pga. "Write a few lines summarizing relevant articles one comes across (which one is likely to refer to in the final report)" — Jims master-skrivingsdokument).

2.1 Taking inspiration from psychology

Essay-materiale.

2.2 Taking inspiration from natural phenomena

The intriguing, diverse, and complex phenomena of nature have for long served as exciting inspirations to human engineers and researchers [ant-colonies, boids, swarms, beeclust]. From taking inspiration when designing a robot traversing one of the most challenging terrains to traverse through, granular surfaces like sand that is, from e.g. zebra-tailed lizards able to run up to 5 meters per second (normally in desert sand) [2], to ... Such phenomena have been subject to considerable study and modelling—and have in fact created entire scientific fields [**bionics**, 3] in which principles from various science- & engineering-diciplines are applied to the physical systems and machines having functions that mimic biological processes, as well as biological processes serving as great sources of inspiration for the engineered systems; *Biomimetics* and *Bionics*, that is. One branch of these types of natural phenomena observed and studied is the biological pulse-coupled oscillators [**russerMinimalAssumptionsReferanser**]. An example of such biological pulse-coupled oscillators found in nature is the synchronously flashing fireflies, as e.g. can be seen in a dark forest in Figure 2.1.

This phenomenon has inspired scientists like Mirollo & Strogatz [5] and in later time Kristian Nymoen et al. [6], to model and replicate this natural phenomenon in human-engineered systems. Given the periodic and repeating nature of the flashing/firing of the fireflies, modelling a firefly has been done by looking at each firefly as a periodic signal or oscillator. This work [5, 6] then ties into the broader work on synchronizing oscillators which has been subject to study for some time now []. What separates Mirollo-Strogatz and K. Nymoen's approaches from these other and previous oscillator-synchronizing

Figure 2.1: Synchronous fireflies at Congaree National Park, United States [1].

methods, is mainly that here the oscillators are *pulse-coupled* (which the fireflies also can be said to be), as opposed to the more "standard" and constraining *phase-coupled* oscillators. Additionally, K. Nymoen's approach accounts for synchronizing initially heterogenous frequencies as well.

2.2.1 Multi-agent systems concepts

- Stigmergy: meaning indirect communication and co-ordination by leaving traces of oneself in ones environment for others to observe or detect subsequently (kan diskuteres).
- Emergence

2.3 Oscillators and oscillator-synchronization

GJØR: [Beskriv dette så godt at du kan snakke fritt om oscillatorers **faser** og **frekvenser** senere (i Implementation f.eks.), spesielt i tilfelle for noen ikke har vært borti det før, eller tatt et Signalbehandlings-kurs].

GJØR: [Skill på Pulse-coupled Oscillators, og Phase-coupled Oscillators].

Teori og nyere cutting-edge/state-of-the-art metoder for fase-/frekvens-synkronisering i oscillatorer og liknende. Se Kristian Nymoens referanser i [6] for eksempel.

Oscillators have been used to implement and model a plethora of systems—also biological—ranging from designing the locomotion-patterns of swimming robot-amphibia through central pattern generators[**Ijspeert**], ..., and as we have already established—modelling synchronously flashing fireflies.

2.3.1 Phase and frequency

Much of the terminology from [6] is used here. An oscillator i is characterized by its *phase* $\phi_i(t)$, which is—at the start of its periodic signal period—initialized to

0 and evolves towards 1 at a rate which is called the *frequency* of the oscillator. So, in mathematical terms, the frequency $\omega_i(t)$ is then given by:

$$\omega_i(t) = \frac{d\phi_i(t)}{dt}. \tag{2.1}$$

When oscillator i's phase is equal to 1 (i.e. when $\phi_i(t) = 1$, or when the periodic signal period is over), we say oscillator i has *phase-climaxed*.

An oscillator-period T is defined as the inverse of the oscillator-frequency ω. In mathematical terms:

$$T = 1/\omega. \tag{2.2}$$

2.3.2 Phase-adjustment

GJØR: [Muligens inkluder introen i Section 5.2 her også (eller bare her?)].

(Hvis relevante og ønskede)
Tidligere metoder til Fase-synkronisering i oscillatorer (puls- og/eller fase-koplede).

2.3.2.1 Mirollo-Strogatz's "standard" phase-adjustment

One approach having been used to achieve this in the past is Mirollo-Strogatz's "Standard" phase-adjustment in oscillators [5].

Each musical agent gets a new phase, $\phi(t^+) = P(\phi(t))$, accoring to the **phase update function** (2.3) upon perceiving a "fire"-event from one of the other musical nodes:

$$\phi(t^+) = P(\phi(t)) = (1 + \alpha)\phi(t), \tag{2.3}$$

where α is the pulse coupling constant, denoting the strength between nodes [6], t^+ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the agent at time t. So, if e.g. $\alpha = 0.1$, then a musical agent's new and updated phase, immediately after hearing a "fire"-signal from another agent, will be equal to $\phi(t^+) = P(\phi(t)) = (1 + 0.1)\phi(t) = 1.1\phi(t)$. 110% of its old phase $\phi(t)$, that is. Hence, and in this way, the agent would be "pushed" to fire sooner than it would otherwise (as nodes fire once they have reached phase-climax $\phi(t) = 1$).

2.3.3 Frequency-adjustment

GJØR: [Muligens inkluder introen i Section 5.3 her også (eller bare her?)].

(If relevant and wanted)
Previous approaches to Frequency-synchronization in oscillators (pulse- and/or phase-coupled) [fixed_freqs, fixed_range_freqs] where the oscillators's frequencies are either equal and fixed, or where frequencies are bound to initialize and stay within a fixed interval/range.

2.3.3.1 Person X's low SA-leveled frequency-adjustment

GJØR: [Beskriv en simplere strategi/metode (to-be-implemented) for å oppnå harmonisk synkronitet i ϕ- & ω-problemet (i.e. problemet der både faser og frekvenser starter med ulike og tilfeldige verdier, og altså begge trenger synkronisering) med frequency-adjustment, da uten noe Self-Awareness-egenskaper, hvis det finnes]**.**

2.4 Musical robots in music technology systems

M. J. Krzyzaniak and RITMO's musical robots, the Dr. Squiggles, have been used in various music technology systems previously [7]. Corresponding 3D-models of these robots were designed by Pierre Potel [8] and will be reused, with permission, for the simulations in this thesis project.

Chapter 3

Baseline

BESKR.: [Mellomkapittel om K. Nymoens approach og teori [6]. Her presenterer jeg teoriene og metodene jeg har brukt mesteparten av masterjobbingstiden på å etterlikne/implementere i Unity].

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

This approach to phase-adjustment works very similarly to the phase-adjustment performed in the "standard" ***Mirollo-Strogatz*** approach presented earlier; the only difference being that now, nodes update their phases with the slightly more complex **phase update function** (3.1) when hearing a "fire"-event from one of the other musical nodes — allowing for both larger, but also smaller, updated phases compared to the old phases:

$$\phi(t^+) = P(\phi(t)) = \phi(t) - \alpha \cdot sin(2\pi\phi(t)) \cdot |sin(2\pi\phi(t))| \tag{3.1}$$

, where t^+ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the agent at time t.

The fact that new and updated phases can both be larger, but also smaller, compared to the old phases, is exactly what's meant by the phase-adjustment being ***bi-directional***, or as the authors call it in the paper as using both excitatory and inhibitory phase couplings between oscillators [6].

The effects then of adjusting phases—upon hearing "fire"-events, according to this newest update-function (3.1)—are that the nodes's updated phases $\phi(t^+)$, compared to their old phases $\phi(t)$, now get decreased if $\phi(t)$ is lower than 0.5, increased if $\phi(t)$ is higher than 0.5, and neither—at least almost—if the phases are close to 0.5. This is due to the negative and positive sign of the sinewave-component in Equation (3.1), as well as the last attenuating factor in it of $|sin(2\pi\phi)| \approx |sin(2\pi\frac{1}{2})| = |sin(\pi)| = |0| = 0$, then if we have $\phi(t) \approx 0.5 = \frac{1}{2}$.

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

This approach to Frequency Adjustment stands in contrast to previous approaches to synchronization in oscillators [fixed_freqs, fixed_range_freqs] where the oscillators's frequencies are either equal and fixed, or where frequencies are bound to initialize and stay within a fixed interval/range.

In order to achieve this goal of *harmonic synchrony* in conjunction with—or rather through—frequency adjustment, we have to go through a few steps to build a sophisticated enough update-function able to help us achieve this.

When it comes to the temporality and timing of when these update functions are used and applied; Musical agents's phases get updated/adjusted immediately as "fire"-/"flash"-events are perceived, whereas agents's frequencies do not get updated until the end of their oscillator-cycle (i.e. when having a phase-climax $\phi(t) = 1$). This is also the reason why frequencies are updated discretely, not continuously. So-called H-values however, being "contributions" with which the frequencies are to be updated according to, are immediately calculated and accumulated when agents are perceiving a "fire"-/"flash"-event — and then finally used for frequency-adjustment/-updating at phase-climaxes.

Each agent i update their frequency, on their own phase-climax (i.e. when $\phi_i(t) = 1$), according to the frequency-update function $\omega_i(t^+)$:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{3.2}$$

where t^+ denotes the time-step immediately after phase-climax, $\omega_i(t)$ is the old frequency of the agent at time t, and $F(n) \in [-1, 1]$ is a quantity denoting how much and in which direction an agent should update its frequency after having received its nth "fire"-signal.

This is how we obtain the aforementioned $F(n)$-quantity:

3.2.1 Step 1: the "in/out-of synch" error-measurement/-score, $\epsilon(\phi(t))$

Describing the error measurements at the n-th "fire"-event, we introduce an Error Measurement function.

The error measurement function (3.3), plotted in Figure 3.1, is calculated immediately by each agent i, having phase $\phi_i(t)$, when a "fire"-event signal from another agent is detected by agent i at time t.

$$\epsilon(\phi_i(t)) = sin^2(\pi\phi_i(t)) \tag{3.3}$$

As we can see from this error-function, the error-score is close to 0 when the agent's phase $\phi_i(t)$ is itself close to 0 or 1 (i.e. the agent either just fired/flashed, or is about to fire/flash very soon). This implies that if it was only short time ago since we just fired, or conversely if there is only short time left until we will fire, we are not much in error or *out-of-synch*.

The error-score is the largest when an agent perceives a "fire"-signal while being half-way through its own phase (i.e. having phase $\phi(t) = 0.5$). We

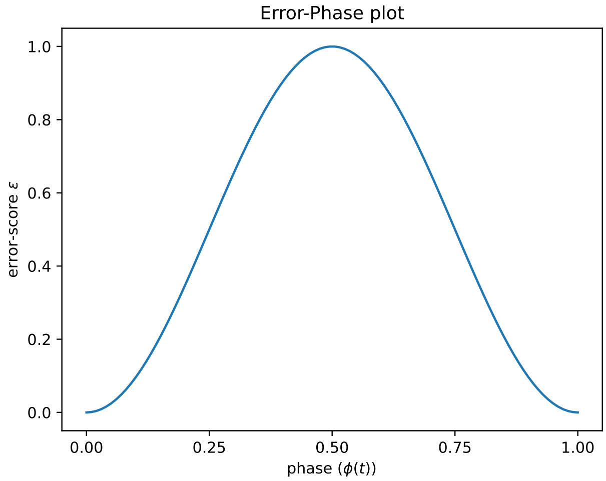


Figure 3.1: Error measurement (3.3) plotted as a function of phase

could also then ask ourselves, does not this go against the main/target goal of the system, being *harmonic synchrony* — if agents are allowed to be "half as fast" as each other? We could imagine a completely "legal" and harmonically synchronous scenario where two agents have half and double the frequency of each other. The agent with half the frequency of the faster agent would then have phase $\phi(t) = 0.5$ when it would hear the faster agent "fire"/"flash" — leading to its Error-score $\epsilon(0.5) = sin^2(\pi/2) = 1$, which then makes it seem like the slower agent is maximally out of synch, when it is actually perfectly and harmonically synchronized. This calls out for an attenuating mechanism in our frequency update function, in order to "cancel out" this contribution so that perfectly harmonically synchronized agents will not be adjusted further despite their high Error-measurement. As we will see below, in Figure 3.2, exactly such an attenuating mechanism is utilized in our frequency-adjustment method.

This error-measurement/-score forms the basis and fundament for the first component of self-awareness, being the *self-assessed synchrony-score* $s(n)$.

3.2.2 Step 2: The first self-awareness component, s(n)

This aforementioned self-assessed synchrony-score, $s(n)$, is in fact simply the median of error-scores ϵ.

If we then have a high $s(n)$-score, it tells us that the median of the k last error-scores is high, or in other words that we have mainly high error-scores — indicating that this agent is out of synch. Conversely, if we have a low $s(n)$-score, indicating mainly low error-scores for the agent — then we have an indication that the agent is in synch, hence leading to low error scores, and in turn low $s(n)$-scores.

In other words, each agent hence has a way to assess themselves in how much in- or out-of-synch they believe they are compared to the rest of the agents. This is then the first degree/aspect of public? self-awareness in the design.

3.2.3 Step 3: frequency update amplitude- & sign-factor, $\rho(n)$

Describing the amplitude and sign of the frequency-modification of the n-th "fire-event" received. It is used to say something about in which direction, and in how much, the frequency should be adjusted.

$$\rho(\phi) = -sin(2\pi\phi(t)) \in [-1, 1] \tag{3.4}$$

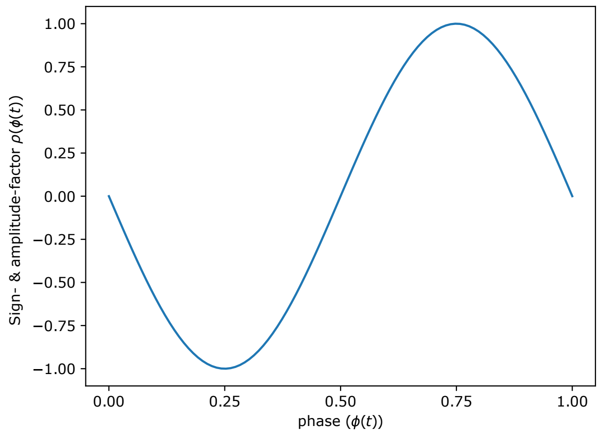


Figure 3.2: The amplitude- & sign-factor, $\rho(\phi(t))$, where $\phi(t)$ is the phase of an agent at time t when it heard "fire"-event n. Notice how the amplitude is equal to 0 when the phase is equal to 0,5.

For example, if an agent i has phase $\phi_i(t) = 1/4$, it gets a value $\rho(1/4) = -sin(\pi/2) = -1$; meaning, the agent's frequency should be decreased (with the highest amplitude actually) in order to "slow down" to wait for the other nodes. Conversely, if an agent j has phase $\phi_j(t) = 3/4$, it gets a value $\rho(3/4) = -sin(3/2\pi) = -(-1) = 1$; meaning, the agent's frequency should be increased (with the highest amplitude) in order to getting "pushed forward" to catch up with the other nodes.

Acts as an attenuating factor, when $\phi(t) \approx 0.5$, in the making of the H-value — supporting the goal of *harmonic synchrony.*

3.2.4 Step 4: the H-value, and the H(n)-list

The following value, being "frequency-update-contributions", is then (as previously mentioned) calculated immediately when the agent perceives another agent's "flashing"-signal:

$$H(n) = \rho(n) \cdot s(n) \tag{3.5}$$

Here we then multiply the factor $\rho(n)$, depicted in Figure 3.2, representing how much, as well as in which direction, the agent should adjust its frequency, together with a factor $s(n) \in [0, 1]$ of the adjusting agent's self-assessed synch-score. This means that all the possible values this $H(n)$-value can take, lies within the green zone in Figure 3.3. We hence see that the smallest value $H(n)$

can take for the nth “fire”-event is -1, which it does when $\phi(n) = 0.25$ and $s(n) = 1$. The highest value it can take is 1, which it does when $\phi(n) = 0.75$ and $s(n) = 1$. We can also see that even though the self-assessed synch-score $s(n)$ (i.e. the median of error-scores) is high and even the maximum value of 1, thus indicating consistent high error-scores (judging by error-function (3.3) and Figure 3.1) — the “frequency-update-contribution” $H(n)$ can in the end be cancelled out, as alluded to before, if in fact the amplitude- & sign-factor $\rho(n)$ is equal to 0. Hence, if we have two agents then where the one is twice as fast as the other, and we accept the $H(n)$-value as the “frequency-update-contribution”, the slower agent which will hear “fire”-events consistently when it has its phase $\phi(n) \approx 0.5$ (if the agents are synchronized) will, even though it gets a high *out-of-synch score* $s(n) \approx 1$, not “be told” to adjust its frequency more by getting a large “frequency-update-contribution”, but in fact “be told” not to adjust its frequency more due to the small or cancelled-out “frequency-update-contribution.”

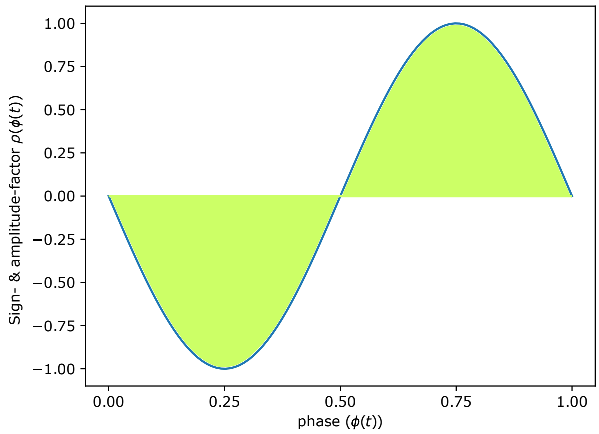


Figure 3.3: All possible $H(n)$-values marked in green, given by $\rho(n) \cdot s(n)$, where $\rho(n)$ is defined as above and depicted in Figure 3.2, and the self-assessed synch-scores $s(n) \in [0, 1]$. The $s(n)$-scores were again simply the median of m error-scores $\{\epsilon(n), \epsilon(n-1), \epsilon(n-2), \ldots, \epsilon(n-m)\}$ which again were given by the error-function (3.3) plotted in Figure 3.1. Observe, in conjunction with Equation (3.6), what the authors [6] point out when describing their frequency update function, when saying that they specify a function that decreases frequency if a fire event is received in the first half of a node’s cycle (i.e. phase $\phi(t) < 0.5$), and speeds up if in the last half (to “catch up’ with the firing node). Also note how the wanted design of the authors, of causing (close to) 0 change to the frequency half-way through the cycle, is enabled by how the “frequency-adjustment-contribution” $H(n)$ can be equal to 0 despite the reacting node having high error scores and a high $s(n)$-score.

To recall, the self-assessed synch-score $s(n)$ tells an adjusting agent how in- or out-of-synch it was during the last m perceived “fire”-/“flash”-events — where $s(n) = 0$ signifies a mean of 0 in error-scores, and $s(n) = 1$ signifies a mean of 1 in error-scores. So then if this H-value is to be used to adjust the nodes’s frequencies with, the frequency will then be adjusted in a certain direction and amount (specified by $\rho(n)$) — given that the agent is *enough* "out of synch"/"unsynchronized" (in the case $s(n)$ is considerably larger than 0).

The H-value says something about how much "out of phase" the agent was at the time the agent's nth "flashing"-signal was perceived (and then followingly how much it should be adjusted, as well as in which direction after having been multiplied together with a sign-factor $\rho(n)$), given then that this H-value also consists of the *self-assessed synch score s(n)* — which again simply was the median of error-scores.

We could look at this H-value as representing the direction and amplitude of the frequency adjustment weighted by the need to adjust (due to being out of synch) at the time of hearing "fire"-/"flash"-event n. Or in other words, this H-value is then the n-th contribution with which we want to adjust our frequency with.

Especially interesting cases are when we have $\phi(n) \approx 0.5 \implies \rho(n) \approx \pm 0$, as well as the last m Error-scores $\epsilon(n)$ being close to 0, also leading to $s(n) \approx 0$. In both of these two cases the entire frequency-adjustment contribution H would be cancelled out, due to harmonic synchronization (legally hearing a "fire"-/"flash"-event half-way through ones own phase) in the first case, and due to not being out of synch in the latter (having low Error-Measurements). Cancelling out the frequency adjustment contribution in these cases is then not something bad, but something wanted and something that makes sense. If these H-values then are cancelled out or very small, it is indicative of that nodes are already in *harmonic synchrony*, and hence should not be "adjusted away" from this goal state. On the other side, if these H-values then are different (e.g. closer to -1 and 1), it is indicative of that nodes are not yet in *harmonic synchrony*, and that they hence should be "adjusted closer" to the goal state.

3.2.5 The final step: the frequency update function, $\omega_i(t^+)$

Now, we can pull it all together, for Nymoen et al.'s Frequency Adjustment approach for achieving harmonic synchrony with initially randomized and heterogenous frequencies.

When an agent i has a phase-climax ($\phi_i(t) = 1$), it will update/adjust its frequency to the new $\omega_i(t^+)$ accordingly:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{3.6}$$

where t^+ denotes the time-step immediately after phase-climax, and $F(n)$ is found by:

$$F(n) = \beta \sum_{x=0}^{k-1} \frac{H(n-k)}{k}, \tag{3.7}$$

where β is the frequency coupling constant, k is the number of heard/received "fire-event"s from the start of the last cycle/period to the end (i.e. the phase-climax, or *now*) — and the rest of the values are as described above.

This $F(n)$-value then, as we see in Equation (3.7), is a weighted average of all the agent's $H(n)$-values accumulated throughout the agent's last cycle.

3.3 System target state: harmonic synchrony

The state of harmonic synchrony is defined [6] 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 []. But somehow, for phases ϕ to be harmonically synchronized, all agent “fire”-signals will have to coincide in a way, even though they don’t all have to incur exactly at the same time always. Exactly how this can look is shown in Section 5.4, especially in Figure 5.5.

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. [6] 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).

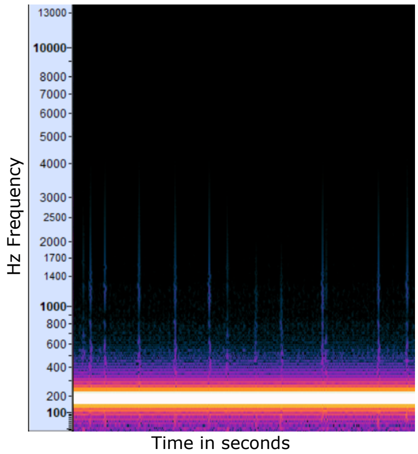
and

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).

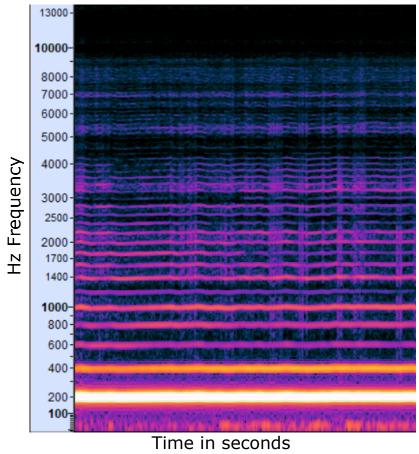
Accomodating for these aspects then, K. Nymoen et al. took inspiration for achieving synchronization in a decentralized system from the concept of *harmonics* in the frequency spectrum of a waveform, in that each harmonic wave or overtone has a frequency with an integer-relationship to the fundamental (smallest) frequency. This phenomenon can e.g. be seen in the frequency spectrogram of a humanly hummed G3-tone, depicted in Figure 3.4b, 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.

More accurately then, and inspired by—although not completely analagous to integer-relationship frequencies like in Figure 3.4b 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.



(a) The frequency spectrogram of the audible waveform being a monotone and purely generated G3-tone at 195,99 Hz [9].



(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 3.4a with my voice.

Figure 3.4: 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 3.4a and 3.4b respectively. Frequencies in a harmonically synchronized agent collective will for the first ϕ-problem resemble the frequencies in 3.4a, 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 3.4b, where these higher frequencies with integer-relationships to the fundamental and lowest frequency can be present.

3.3.1 Detecting harmonic synchrony

In order to test and evaluate synchrony-performance in their firefly-inspired oscillator-system, K. Nymoen et al. [6] 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 5.5 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 [6], 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 4

Tools and software

BESKR.: ["which describes what you've used" — ish Kyrre].

BESKR.: [Kan flyttes til en egen seksjon hvis dette kapittelet ikke ville vært så stort (jf. 'ThesisChecklist' på Robin-wikien)].

BESKR.: [(Hentet fra Tønnes sin master, om Tools and engineering) En introduksjon til de forskjellige verktøyene og prosessene brukt iløpet av masteroppgaven. Fokuser på fysisk arbeid gjort, og ingeniør-delene av masteroppgaven, inkludert 3D-design av de fysiske robotene, valg av deler, simulering i systemer, og testingen, valideringen, og verifikasjonsmetoder brukt i oppgaven. Gjerne også en oversikts-tabell av verktøy og programvare brukt].

Software:

- Notepad++ v8.1.9.2 (64 bit). For writing my master's thesis and code.
- Unity Version 2021.2.0f1. Unity is originally a game-development platform, but can also be used to make **INKL.:** [realistic]? simulations containing physical rigid-bodies using the ¡bla.bla Rigidbody¿-physics engine.
- Python 3.10.0. for plotting and analysing data.
- Inkscape 1.1.1 for editing and creating vector graphics.
- Gimp 2.10.30 for editing and creating raster graphics.
- Audacity 3.1.0 for plotting frequency-spectra of recorded waveforms in Figure 3.4.

Hardware:

- CXT USB-Microphone for recording the hummed waveform, as in Figure 3.4b.

Chapter 5

Implementation

BESKR.: [Her presenterer jeg re-implementasjonen/etterlikningen/implementasjonsspesifikke-valg av K. Nymoens approach og teori i det nye systemet/simulatoren min i Unity — samt hvordan jeg har verifisert at mekanismene fungerer (e.g. fase-synkroniseringsplott)].

This chapter gives an overview of the developed musical multi-robot system, methods implemented for it, as well as the performance measure used to evaluate these methods. The main goal of the implemented system is to allow for individual musical agents in a musical multi-agent collective to interact with each other, in order to achieve emergent and co-ordinating behaviour—in our case synchronization—with varying degrees of self-awareness, collective-sizes, and of difficulty and certainty in the environment and communication. More specifically, the goal with the design is to enable the robot collective to achieve so-called *harmonic synchronization* within a relatively short time. Exactly what is meant by *harmonic synchronization* will be expounded in Section 3.3.

These goals firstly require of the agents the modelling of oscillators with their properties, like phase and frequency, as explained further in Subsection 5.1.1. To allow for interaction and communication between the agents, mechanisms so that the agents can transmit "fire"-signals, as well as listen for other agents's "fire"-signals, is necessary as well, and is presented in Subsection 5.1.2.

First, the system and the system components will be presented and introduced. Then, methods implemented for achieving the system target goal of *harmonic synchrony* in various synchronization objectives—firstly solely for oscillator-phases, then secondly for both oscillator-phases and oscillator-frequencies—will be described and presented. How the system target state of harmonic synchrony is detected will then be described in Section 5.4.

5.1 Simulator setup: the musical multi-robot collective

BESKR.: [Introduserer og presenterer det utviklede (simulator-)systemet du har utviklet i Unity selv, veldig gjerne med et fint Unity-/Simulator-systemskjema].

Envision that we have a decentralized (i.e. no central control) multi-agent collective scenario consisting of musical robots modelled as oscillators. These are solely communicating through brief "fire"-like audio-signals—greatly inspired by K. Nymoen et al.'s synchronizing "fireflies" [6]. They are not initially synchronized in their firing of audio-signals; but as time goes, they are entraining to synchronize to each other by adjusting their phases and frequencies when/after hearing each other's audio-/fire-signals. If they then, after some time of listening to each other and adjusting themselves accordingly, succeed in becoming synchronized — we then will eventually see "fire"-events/-signals line up at an even underlying pulse or rhythm. Examples and demonstrations of this process are depicted in Figure 5.1 and Figure 5.2.

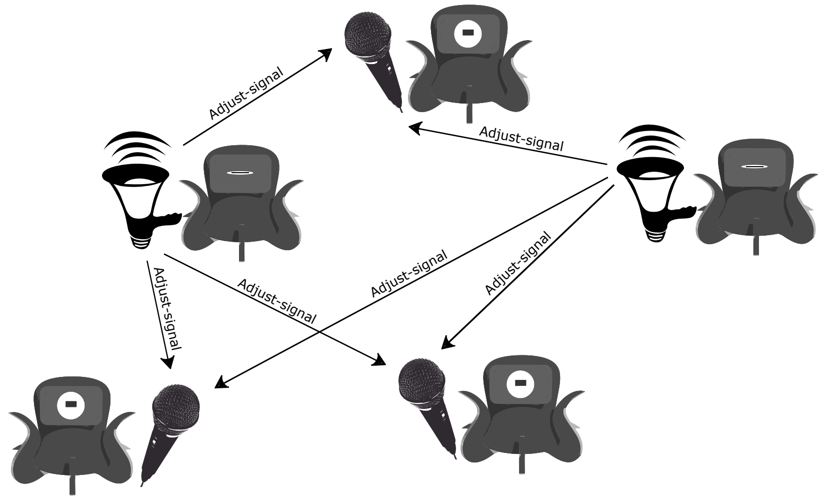
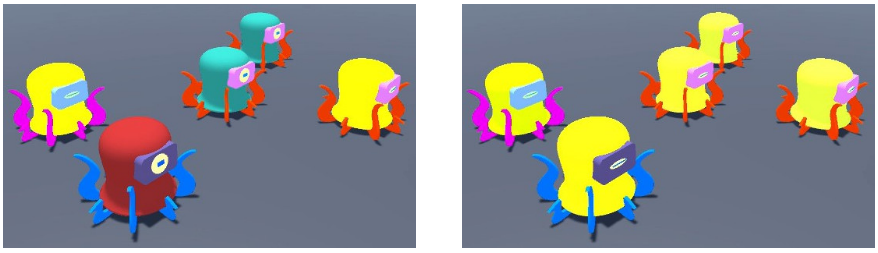


Figure 5.1: Illustration/Schematic: The musical robot collective entraining to synchronize to each other, or more specifically to achieve harmonic synchronization, through performing phase- & frequency-adjustments. Agents that are not firing at the moment will adjust themselves after hearing a transmitted "fire"-/adjustment-signal from a neighbouring firing agent.

5.1.1 The individual agent: a musical robot

As introduced and presented earlier, our musical robot collective will then consist of models of M. J. Krzyzaniak and RITMO's *Dr. Squiggles*, 3D-models of which can be seen in Figure 5.2.

Every musical robot or node have certain components, attributes, and characteristics that make it what it is. Such include an oscillator-component, consisting of the agent's oscillator-phase ϕ and oscillator-frequency ω. Notions like "agent", "robot", "firefly", and "oscillator" will be used interchangeably throughout the thesis. The agents have an input-mechanism for hearing/detecting transmitted "fire"-event signals from other agents, as well as an output-mechanism for transmitting or playing such "fire"-/adjust-signals or tones, as is illustrated



(a) Screenshot from the very beginning of a Synchronization-simulation.

(b) Screenshot from the same Synchronization-simulation as in 5.2a, a few moments later.

Figure 5.2: From simulation: An example of the system target goal of harmonic synchrony being achieved in a musical robot collective, where oscillator-frequencies ω are constant and equal (1Hz), or in other words synchronized already, but oscillator-phases ϕ are unsynchronized and initially uniformly random numbers in the range of $[0, 1]$.
At first in 5.2a, we see the agents firing, i.e. blinking with their eyes and turning their body yellow, asynchronously at first. Only two robots, one with pink and one with red tentacles, fire synchronously so far. Seconds later in 5.2b, after having listened to each others's adjustment-signals and adjusted themselves accordingly, all agents now fire simultaneously and synchronously.

with the microphone and megaphone respectively in Figure 5.1.

In order to be able to analyse the musical scenario within which they are situated (self-assessment), as well as for adapting their musical output accordingly (self-adaptation), the agents are to some extent endowed with artificial intelligence and self-awareness capabilities. The robots are self-aware of their own phase and frequency, but are unaware of other agents's true phases and frequencies. They also possess the self-assessment capability of evaluating how much in- or out-of-synch they are, as seen in the greater context of the entire robot collective. When the agents hear the transmitted "fire"-/adjust-signals, the agents are intelligent enough to adjust themselves in the direction of the system goal/target state.

5.1.2 Robot communication: the "fire"-signal

These aforementioned audio-signals, also referred to as "fire"-signals, "flash"-signals, or adjust-signals, are transmitted whenever an agent's oscillator *peaks* or *climaxes* (i.e. after its cycle or period is finished, having phase $\phi(t) = 1$) — or actually after every second *peak*, as a way (discovered by K. Nymoen et al. [6]) to attain the system target goal of *harmonic synchrony*, to be elaborated upon in Section 3.3.

The "fire"-signals are short and impulsive tones that the agents output through their loudspeakers. These short audio-signals/sounds "wildly" transmitted or played into the environment are then the only means of communication within the multi-agent collective, implying that are agents are pulse-coupled, not phase-coupled, oscillators. In other words, our agents will communicate and coordinate with each other through the very typical multi-agent system concept of *stigmergy*.

When an agent detects a “fire”-/adjust-signal, the agent will adjust its own oscillator-properties (phase ϕ and frequency ω), depending on which type of problem the agents are to solve. No individual agent is directly able to adjust or modify the state or properties of any other agent, only its own. Exactly the type of problems we attempt to solve in this thesis will be presented now in Section 5.2 and Section 5.3.

5.2 Synchronizing oscillator-phases

GJØR: [Introduser det første ϕ-problemet, uthevet og i fet skrift. Deretter fortsett til løsningene dens (Phase-Adj.-metodene). "This is the first and simpler problem to solve, namely synchronizing the phases ϕ_i of all agents i."].

If we first assume constant and equal oscillator-frequencies in our agents, we can take a look at how the agents adjust their—initially random—phases in order to synchronize to each other. We will from here on and out refer to this first problem as **the ϕ-problem**, given that the phases (ϕ_i) for all agents i are what we need to adjust and synchronize — and that frequencies (ω_i) technically already are synchronized.

The goal state of the agents is now for all agents to fire/flash simultaneously, after having started firing/flashing at random initially. Note that this is a special case of the final and ultimate goal of *harmonic synchrony*. This is due to how all agents in the collective firing/flashing simultaneously, is considered having achieved harmonic synchrony since its phases would be synchronized if fire-events are lined up in even pulses, as well as all frequencies in the agent collective being within the set of “legal” frequencies, $\omega_0 \cdot 2^{\mathbb{N}_0}$, where ω_0 is the fundamental (smallest) frequency in the agent collective, and $0 \in \mathbb{N}_0$ — leading to $\omega_0 \cdot 2^0 = \omega_0$ to be a legal frequency, which is what all agents in our case here have as frequences.

In order for the musical agents to synchronize to each other, they will have to—due to their heterogenous and randomly initialized phases—adjust or update their own phases according to some well-designed update-/adjustment-functions, as presented below.

When it comes to the temporality and timing of when these updating functions are used and applied; Musical agents's phases get updated/adjusted immediately as “fire”-/“flash”-events from neighbouring robots are perceived.

5.2.1 Verifying Mirollo-Strogatz's phase-adjustment

Mirollo-Strogatz's approach for synchronizing phases in oscillators, as introduced in 2.3.2.1, is implemented in the Unity simulator, and each agent is endowed with **phase update function** (2.3) with which they adjust themselves according to when perceiving a “fire”-signal as described above.

The verification that this works in the newly built synchronization-simulator was performed by dumping all agents's phase-values $\phi(t)$ during simulation-runs. A plot of these $\phi(t)$-values, evolving through simulation-time in seconds, is shown in Figure 5.3.

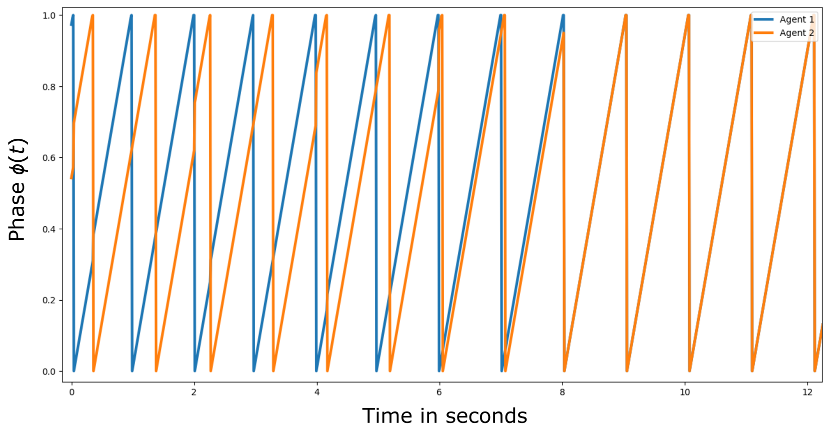


Figure 5.3: “Standard” phase-adjustment with Mirollo-Strogatz’s approach

5.2.2 Verifying K. Nymoen’s bi-directional phase-adjustment

K. Nymoen et al.’s approach for synchronizing phases in oscillators, as introduced in Section 3.1, is implemented in Unity, and each agent is endowed with **phase update function** (3.1) with which they adjust themselves according to when perceiving a “fire”-event as described above.

The verification that this works in the newly set-up simulator-environment was performed by analysing carefully all the agents’s phase-values $\phi(t)$ throughout a simulation-run. Such an analysis/plot can be seen in Figure 5.4.

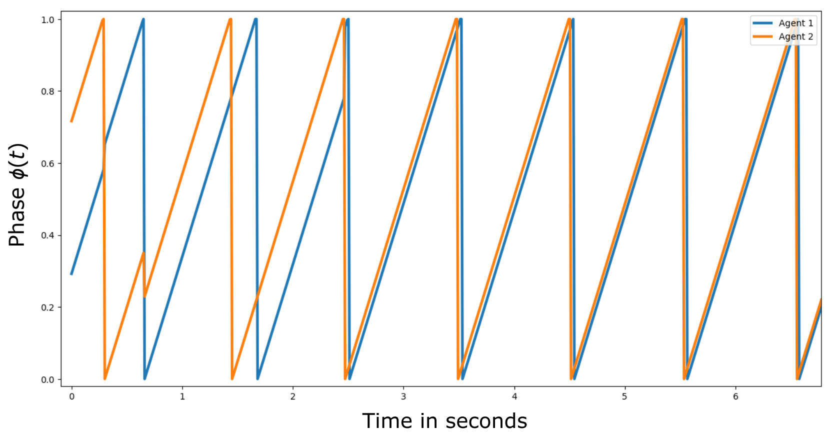


Figure 5.4: Bi-directional phase-adjustment with K. Nymoen et al.’s approach

5.2.3 Thorvaldsen’s self-aware phase-adjustment

GJØR: [Beskriv min nye proposede algoritme (to-be-implemented) for å oppnå harmonisk synkronitet i ϕ-problemet med phase-adjustment, som inneholder flere tilleggs- Self-Awareness-komponenter, sammenliknet med K. Nymoens bi-directional phase-adjustment metode.

Eksempler på slike tilleggs- Self-Awareness-komponenter er såkalt Belief-awareness (som fanger usikkerhet og tillits-nivåer) og/eller Expectation-awareness (som kombinerer Belief-awareness og Time-awareness) [4]. Andre forslag er å implementere Self-Awareness i forhold til:

- avstand: f.eks. $H^*(n) = H(n) \cdot \frac{1}{distance}$ for "fire"-event n, så lenge ikke $distance = 0$.
- hvem som er hvem (i.e. agent_id).

].

5.3 Synchronizing oscillator-frequencies

GJØR: [Introduser det andre, og mer utfordrende, ϕ-&ω-problemet, uthevet og i fet skrift. Deretter løsningene dens (Freq.-Adj.-metodene), som man må bruke i tillegg til ϕ-løsningene eller Phase-Adjustment-metodene. "This is the second and harder problem of synchronizing both phases ϕ_i, as well as frequencies ω_i, for all agents i."].

When we open up for the possibility for heterogenous frequencies in our musical agent collective, we open up to exciting musical aspects like the playing of diverse rhythmic patterns as e.g. mentioned in Section 3.3, but we then also need to not only synchronize phases, but also frequencies, simultaneously. This second problem of adjusting synchronizing both all phases ϕ_i and frequencies ω_i, the values of which can all be heterogenous and initially random, for all agents i in the agent collective — we will from here on and out refer to as **the ϕ-&ω-problem**. This slightly more complex problem calls for us to also find solutions on how to adjust and synchronize frequencies. We already have methods by which we can adjust and synchronize the agents's phases ϕ with, described in Section 5.2, but we do so-far lack methods by which we can adjust and synchronize the agents's frequencies ω with.

We hence now introduce randomly initialized, non-constant, and heterogenous oscillator-frequencies in our musical agents. The agents are now required to synchronize their initially different and random frequencies, so that frequencies are "legal" and *harmonically synchronized*. Such "legal" frequencies are described clearly in detail in Section 3.3.

Some implemented approaches for achieving this are presented now. Notice the increasing degree of *Computational Self-Awareness* endowed in the methods.

5.3.1 Verifying K. Nymoen's frequency-adjustment

In the newly proposed Unity simulator environment, the previously introduced self-assessed synch-score $s(n)$ (in 3.2.2) is implemented as a list containing m error-scores ϵ. Such a list is easily implemented in C# by declaring a List<float> called *errorBuffer* e.g. (i.e. *errorBuffer* is a list containing floating point values):

$$errorBuffer = \{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}, \tag{5.1}$$

then leading to:

$$s(n) = median(errorBuffer) = median(\{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}) \in [0,1], \tag{5.2}$$

where n is the latest observed "fire-event", and m is the number of the last observed "fire"-events we would like to take into account when calculating the self-assessed synch-score.

Regarding the "frequency-update-contributions" (the H-values described in 3.2.4) in my Unity-simulator, all the calculated H-values are accumulated and stored in an initially empty C#-list (of floats), referred to as $H(n)$, at once they are calculated. The $H(n)$-list is then consecutively "cleared out" or "flushed" when its H-values have been used for the current cycle/period's frequency adjustment (i.e. at the phase climax, when $\phi(t) = 1$), and is then ready to accumulate new H-values during the next cycle/period.

5.3.2 Thorvaldsen's high SA-leveled frequency-adjustment

GJØR: [Beskriv min nye proposede algoritme (to-be-implemented) for å oppnå harmonisk synkronitet i ϕ- & ω-problemet med frequency-adjustment, som inneholder flere tilleggs- Self-Awareness-komponenter, sammenliknet med K. Nymoens frequency-adjustment metode. Eksempler på slike tilleggs- Self-Awareness-komponenter er såkalt Belief-awareness (som fanger usikkerhet og tillits-nivåer) og/eller Expectation-awareness (som kombinerer Belief-awareness og Time-awareness) [4]. Andre forslag er å implementere Self-Awareness i forhold til, og vekte frekvensoppdaterings-bidragene H(n) i henhold til:

- avstand: f.eks. $H^*(n) = H(n) \cdot \frac{1}{distance}$ for "fire"-event n, så lenge ikke $distance = 0$.
- hvem som er hvem (i.e. agent_id).
- større median-liste ift. den self-assessed'e synch-score'n $s(n)$ og $errorBuffer$'et. Dette kan være nyttig ved større collective-sizes, da et lite/kort median-filter/-$errorBuffer$-liste vil kunne miste eller gå glipp av error-scores, $\epsilon(n)$, fra "fire"-events fra langt tilbake (tidlig) i "oppsamlings-perioden."
- de andres frekvenser. Høre etter og registrere andre individers "fire"-signaler kontinuerlig og estimere disse individenes frekvenser utifra det (f.eks. $\hat{\omega}_j = time_{j,fired\_now} - time_{j,fired\_last\_time}$, eller et gjennomsnitt av slike oppsamlede verdier).

].

5.4 Performance-measure: time until harmonic synchrony is detected

GJØR: [Legg inn gode forklaringer på hvordan du implementerte performance-/synch-measuret ditt (vha. **Synchrony Perf.-measure** reMarkable-notatet, schematics (som skissene mine), matte-uttrykk, og evt. algoritmer)].

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 5.5. 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 [6]). 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 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 5.6.

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?)

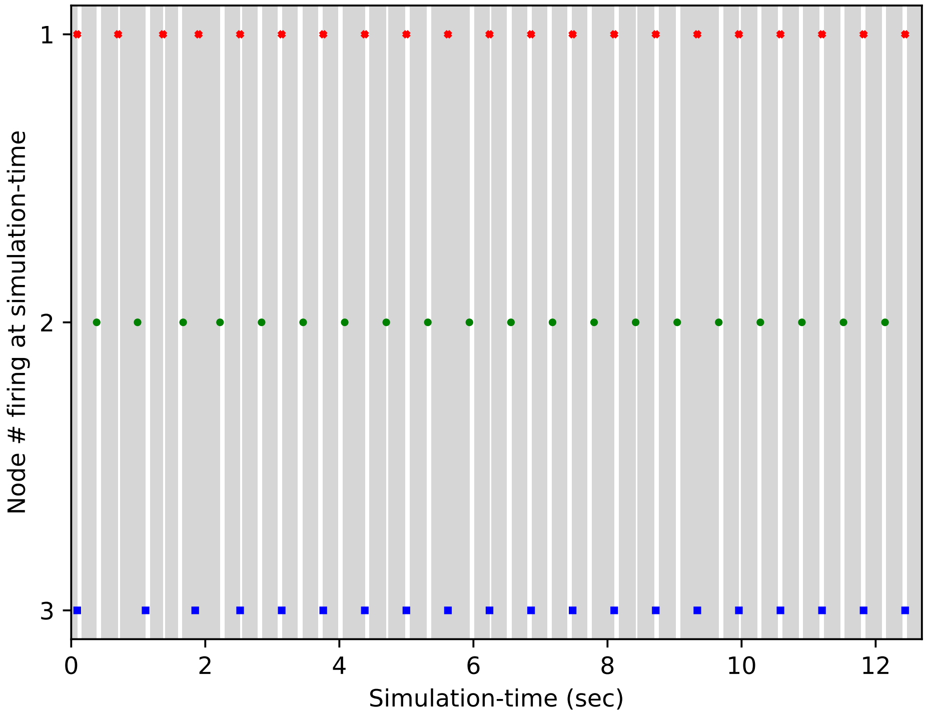


Figure 5.5: A **performance-measure plot**, displaying the temporally recorded pulses/"firings" transmitted by the robots in the Dr. Squiggles-collective throughout the synchronization-simulation in Unity. Short and white windows/strips in the figure represent the short 'legal firing' time-periods t_f 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).

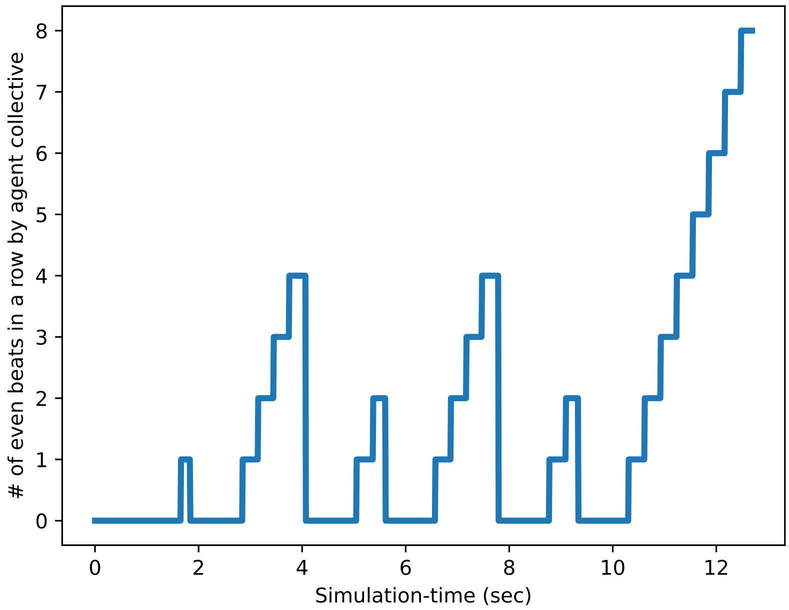


Figure 5.6: A **synchrony-evolution plot**, displaying the temporal recording of the *towards_k_counter*-variable throughout a synchrony simulation-run in Unity. The counter is incremented as the robot-collective fires evenly within 'legal' t_f-windows, and is conversely reset to 0 if illegal firings during 'silent' t_q-windows are heard. Note that in this specific simulation-run above (same run as in Fig. 5.5), 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.

Chapter 6

Experiments and results

BESKR.: [Et kapittel der du har satt opp (med hyper-parametere og miljøvariabler f.eks. i en oversiktlig tabell) eksperimenter, kjørt simulation-runs med disse verdiene, og viser hva resultatene ble. Resultater kan være **performance-plot**s (ift. harmonic synch.-times for diverse hyper-parametere og miljøvariabler, der plottet må lages spesifikt for hvert eksperiment, og kan f.eks. være box-plot eller tabeller med performance-/hsynchtime-/simulation-time(s)-verdier), **performance-measure plot**s (altså hvordan harmonic synch. ble detectet i et simulation-run, med *plot_PerformanceMeasurePlot_for_SimRun.py*), **phase-&frequency-plot**s (altså hvilke fase- og frekvens-verdier robotene hadde iløpet av simulation-run'et, via *plot_PhaseFrequencyPlot_for_SimRun.py*), eller **synchrony-evolution plot**s (der 'towards_k_counter'en iløpet av simulation-run'et plottes, via *plot_SynchronyEvolutionPlot_for_SimRun.py*)].

6.1 Solving the simpler $\phi-$problem

This is the section for the experiments attempting to solve the first and simpler problem, namely synchronizing the phases ϕ_i of all agents i.

6.2 Solving the harder $\phi\&\omega-$problem

This is the section for the experiments attempting to solve the second and harder problem of synchronizing both phases ϕ_i, as well as frequencies ω_i, for all agents i.

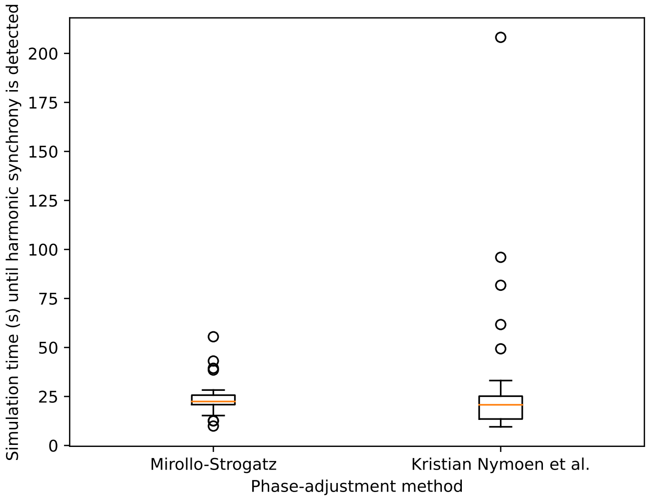


Figure 6.1: Performance-plot: harmonic synchronization-times from initial simulator-experiment when synchronizing phases ϕ_i for all agents i, where all phases are initially uniformly randomized between 0 and 1, and eventually synchronize and align. We here measure how long it takes 6 agents to synchronize their phases to each other, given the two different phase-adjustment methods. 30 individual runs per phase-adjustment method were performed in Unity for a collective-size of 6 agents, and $\alpha = 0.2$ e.g.

Chapter 7

Conclusions

BESKR.: [where I shall follow-up on my *research questions* by a discussion of to which degree—and in what ways—the thesis-/project-work has answered them].

GJØR: [Se på (for kapittel-inspirasjon):

- Tønnes . MSc-thesis . Discussion-Ch.
- Jim . 'how to write a master thesis.pdf' . 'Conclusions'.

].

Bibliography

[1] @columbiasc and @_flashnick. *RoadTrip: Synchronous Fireflies at Congaree National Park*. URL: https://avltoday.6amcity.com/synchronous-fireflies-congaree-national-park/ (visited on 01/26/2022).

[2] Daniel Goldman, Haldun Komsuoglu, and Daniel Koditschek. "March of the sandbots". In: *IEEE Spectrum* 46.4 (2009). Publisher: IEEE, pp. 30–35.

[3] nature journal. *Biomimetics*. URL: https://www.nature.com/subjects/biomimetics (visited on 03/30/2022).

[4] Peter Lewis et al. "Towards a Framework for the Levels and Aspects of Self-aware Computing Systems". In: *Self-Aware Computing Systems*. Ed. by Samuel Kounev et al. Cham: Springer International Publishing, 2017. Chap. 3, pp. 51–85. ISBN: 978-3-319-47474-8. DOI: 10.1007/978-3-319-47474-8_3.

[5] Renato E. Mirollo and Steven H. Strogatz. "Synchronization of pulse-coupled biological oscillators". In: *SIAM Journal on Applied Mathematics* 50.6 (1990). Publisher: SIAM, pp. 1645–1662.

[6] 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.

[7] Silje Pileberg. *Say hi to the musical robot "Dr. Squiggles"*. URL: https://www.uio.no/ritmo/english/news-and-events/news/2021/drsquiggles.html (visited on 02/02/2022).

[8] Pierre Potel. *SoloJam-Island*. URL: https://github.com/67K-You/SoloJam-Island (visited on 02/02/2022).

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