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

Autumn 2021

Computational Self-Awareness in Musical Robotic Systems

David Thorvaldsen

Abstract

GJØR: [Tenk igjennom og besvar så godt som mulig etterhvert:

- What Is *Self-Awareness*, according to some of history's most prominent thinkers, including central philosophers and psychologists? Or first things first, what Is *the self*? And what exactly is *awareness*? What implications does this have as it pertains to the domain and practice of computer science and engineering — especially when designing and describing computing systems? What lessons can designers and researchers of computing systems learn, what can they be inspired by, and which computational benefits/advancements in engineering can be made when exploring the answers to these questions?

].

INKL.: [Despite its elusive, and at times hard-to-tangibly-define, nature; the psychological concept of Self-Awareness have relatively recently served as a rich source of new inspiration, previously unknown or undiscovered for the computer engineer.]**?**

Acknowledgement(s$^?$)

Hei og takk til alle dere som støttet og hjalp meg igjennom denne reisen.

Contents

List of Tables

List of Figures

GJØR: [Finn ut hvordan man endrer Figure-navnene her til noe annet enn det de har som caption i hoved-teksten].

Chapter 1

Introduction

1.1 Motivation

GJØR: [Tenk igjennom og besvar så godt som mulig etterhvert (kompilert fra Samuelsens MSc-thesis, og etterhvert Essay-kommentarer. Kan cross-checke med Tønnes og hun andres også):

- **Why is the thesis topic, and its outflowing proposed solutions/improvements, of relevance in the world today?**
  - History of field, how things have been done before — and why the situation/needs/requirements might have changed, or why these traditional/typical solutions may be ripe for improvements or better solutions? Why are these concerns/problems/factors of importance?
    - Demonstrate, illustrate, and explain these changes / this new situation so that the reader understands why your topic's contributions are necessary or needed.
  - What are the relevant real-world problems in need of solutions/improvement, where the thesis topic can provide such solutions/improvements?
  - Differentiate between what the "Background-/Related-works-proposed method" conributes with, and the "new proposed method" that you yourself want to try out (e.g. differentiate between ODA-loops and MAPE-K-loops, and endowing computational systems with *computational self-awareness* (and *self-expression*).
    - Explain why the "new proposed method" is needed/granted, maybe in relation to a lack or challenge with the original "Background-/Related-works-proposed method". Perhaps also mention the absence or "freshness" of this "new proposed method" in the history or field of the "Background-/Related-works-proposed method".

    "From Essay-comments": [

  - What constitutes the essential ideas behind the solutions to the problem?

- Have I written a short description of the problem / challenge for my thesis?
- Have I introduced (understandably and intuitively) Self-Awareness as an exciting and relevant field/source-of-inspiration-and-concepts?
- Have I presented motivations and arguable advantages of endowing a computational system with Self-Awareness (Se reMarkable'n)?
- Have I discussed or argued for the motivations for endowing, especially and in particular, music systems with *Computational Self-Awareness* — and then connected this with efforts like Nymoen et al.'s Firefliy-synchronization and/or Chandra et al.'s Solojam?].

].

"Fra Essay-Introduction": [

INKL.: [Designing and predicting all possible scenarios of a computing system at design-time is often hard, and sometimes impossible (in the case of unpredictable faults e.g.). If one wants to achieve coordination and continuous adaptation of a system or of system-components (for example in a collective) – some sort of intelligence might be necessary to endow it with. Endowing computing systems with Self-Awareness can be beneficial in several respects, including a greater capacity to adapt, to build potential for future adaptation in unknown environments, and to explain their behaviour to humans and other systems [SACS 17 Ch. 3]. Self-awareness concepts from psychology are inspiring new approaches for engineering computing systems which operate in complex dynamic environments [SACS 16 Ch. 2]. As we can see in various Music Technology Systems, this endowing can also give rise to interesting cooperative and coordinating behaviour.]**?**

INKL.: [The problem of this thesis will mainly consist of studying the effects differing Self-Awareness levels, varying collective-sizes, levels of task difficulty (like more complex behaviours, and limited communication) – have on usefulness, system dynamics, overall performance, and scalability (primarily within the domain of Musical Multi-Robot/-Agent collectives).]**?**].

"Fra Essay-Introduction-kommentarer": [

INKL.: [In this MSc. thesis, we will explore an exciting and relatively new translation of the concepts and notions regarding *self-awareness* – as they pertain to humans and animals especially – from the domain of Psychology, into the domain of Computation and Engineering.]**?**

INKL.: [We will in this project attempt to implement and explore whether, and indeed in what way (how), a computational system (like computational agents/agent-collectives in a computer-simulation, and/or even on physical musical robots) can exibit and display Self-Awareness (and corresponding Self-Expressive) capabilities, comparable to how they are perceived in humans and animals.]?

INKL.: [We will in this thesis project investigate methods and models for computational self-awareness in multi-robot systems, with application to the musical robotics domain.]?].

INKL.: [Engineering a computing system for a certain environment often requires some knowledge of said environement — both on the end of the creator of the computing system, as well as for the computing system in turn. This is at least the case in autonomous computing, where computing systems are supposed to be able to observe, learn, adapt, and act on their own — independently from their creator.]?

However, predicting all possible future states of complex, dynamic, and ever-changing environments is hard, and at times impossible. **INKL.:** [This calls for online and continuous learning, don't you think? How to best tackle this problem? Glad you asked. — With Self-Awareness of course. Because ...]?

GJØR: [Summer de viktigste bidragene av arbeidet her eller under 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

GJØR: [Kople tekst oppmot (de 2-3) *research-questions*'a mine her].

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

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 to synchronize to each other efficiently despite these difficulties/challenges?

1.3 Outline

GJØR: [Skriv opp strukturen/oversikten (Eagle's-eye) av thesis-dokumentet her]**.**

Chapter 2

Background

GJØR: [Skriv opp bulletpoints fra mulige inspirasjoner og referanser her. "Write a few lines summarising relevant articles one comes across (which one is likely to refer to in the final report)" - Jims master-skrivingsdokument)].

"Fra Essay-kommentarer": [

INKL.: [Often times, scientists have drawn inspiration from various scientific fields – particularly different fields from ones own – into their own field, for various reasons. Indeed, the translated concepts (from the one domain to the other) will most likely be accompanied with brand new ways to think about ones own domain, as well as other domains again interacting with it (hence having a real opportunity to start a "domino"-like chain-reaction of new ways to think about things emerging). These new ways to think about things (often in ones own domain) – apart from being interesting and intriguing – might be useful, both for ones own field but also for other fields again (especially if thinking long term). For example in the Multi-Agent Systems (MAS) field, it has been a common practice to study complex biological systems in nature, in order to translate these mechanisms into the technology- and engineering-domain (be it the *Ant Colony(?)*, or *Beeclust(?)*). Such bio-inspired algorithms have been – *and still are(?)* – some of the most widely used optimization algorithms throughout history.]**?**

INKL.: [(Fra Essay om 'path planning', 'EA's og 'multi-objective optimization'. Definitivt ikke kopier, men skriv om isåfall): "This essay attempts to give an overview over the fields of path-planning, evolutionary algorithms and multi-objective optimization, including pointers to recent work in these fields, especially where they intersect. In tradition with other literature relating to evolutionary algorithms, algorithms which are not population-based or otherwise based on principles similar to evolutionary algorithms are called 'classical' to separate them from evolutionary variants."]**?**].

2.1 Nymoen et al. sin Firefly-Synkronisering:

INKL.: [
Nymoen et al. [1] showed how one can, by endowing musical agents with self-awareness capabilities, achieve *harmonic synchrony* of phases and frequencies in pulse-coupled oscillators.

"Sigmund Kjøkken-Recall": [

BESKR.: [Kanskje denne seksjonen blir gjort overflødig av en god forklaring på Oscillatorer og synkronisering av Oscillatorer i 'Background'-kapittelet, og en god forklaring av akkurat hvordan dette oppnås i 'Implementation'-kapittelet?].

GJØR: [Demonstrer/Illustrer poenget bak fingrene og tidsaksen på bordet (ish det som er i figuren under for fase), og så det samme for frekvens-justering med f.eks. halve—eller noe annet—som start-frekvens; og at de da ender i *harmonisk synkroni*].

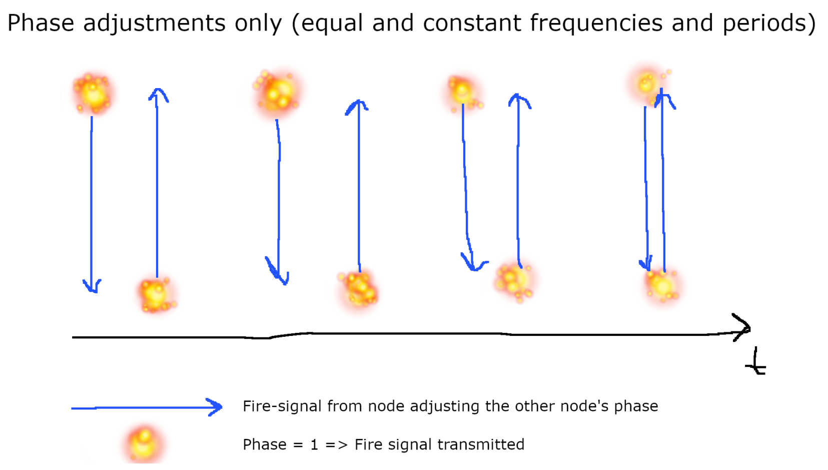

Phase adjustments only (equal and constant frequencies and periods)
t
Fire-signal from node adjusting the other node's phase
Phase = 1 => Fire signal transmitted

Figure 2.1: Fase-justering (som om man skulle tappet med fingrene på et bord)

].

"Tante-Kjersti-inspirert (ish Stue-Recall)": [
The diverse and complex phenomena of nature have for long served as exciting inspirations to human engineering and research (cite ant colonies, boids & swarms, beeclust e.g.). One such phenomena studied and attempted modelled is the synchronous firing of fireflies in the rainforests.

GJØR: [Insert illustration/picture of synchronizing/synchronized fireflies firing in a dark forest here].

This has inspired scientists like Mirollo & Strogatz [], and in later time Kristian Nymoen, Kyrre Glette et al. [1], to attempt to model and "etterlikne" this natural phenomenon in human-engineered systems. This work ties into the work on synchronizing oscillators []$^?$ which has been subject to study for some time now. What separates Mirollo & Strogatz and K. Nymoen's approach from these previous ones, is that here the oscillators are *pulse-coupled*, as opposed to the more normal and constraining *phase-coupled* (explain$^?$). Each modelled "firefly", or firing node, is here implemented and considered as an oscillator, characterized by its phase and frequency. **INKL.:** [Kinda, the job is to align sinusoidal waves, either by shifting an agent's phase "up", or "down".]**?**

INKL.: [No training of any neural networks or any model-data was needed to achieve synchrony in this case — and so far no machine learning is used — but instead we see an emergent *harmonic synchrony* in a collective, by endowing fairly simple agents with not too complicated update-functions. This is well known in the Multi-Agent Systems & Swarm Robotics literature []$^?$.]**?**]**.**
]**?**

2.2 Oscillators and Oscillator-Synchronization

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

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

Much of the terminology from [1] is used here. An oscillator i is characterized by its *phase* $\phi_i(t)$, which is initialized to 0 and evolves towards 1 at a rate of $\frac{d\phi_i(t)}{dt}$ — which is also called the *frequency* of the oscillator.

2.2.1 Phase-adjustment

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.

The goal state of the agents is now to fire/flash more or less simultaneously, after having started firing/flashing at random initially.

One approach having been used to achieve this in the past: Mirollo-Strogatz "Standard" Phase-adjustment in oscillators [], as in Figure 2.2.

Each musical node updates its phase ϕ accoring to the **phase update function** (2.1) when hearing a "fire"-event from one of the other musical nodes:

$$P(\phi) = (1 + \alpha)\phi, \tag{2.1}$$

where "*α is the pulse coupling constant, denoting the strength between nodes*" [1]. So, if $\alpha = 0.1$ e.g., then a musical node's new and updated phase, im-

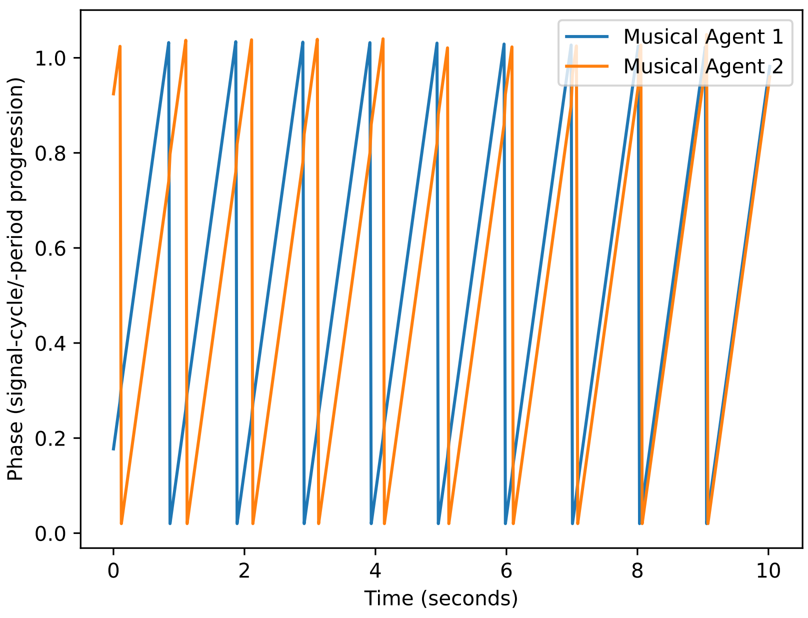


Figure 2.2: "Standard" Phase-Adjustment with Mirrollo-Strogatz's approach

mediately after hearing a "fire"-signal from another node, will be equal to $P(\phi) = (1 + 0.1)\phi = 1.1\phi$. 110% of its old phase ϕ, that is. Hence, and in this way, the node would be "pushed" to fire sooner than it would otherwise. An illustration of this is given in Figure 2.2.

2.2.2 Frequency-adjustment

* previous approaches to synchronization in oscillators cite(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. *

Chapter 3

Tools and Software

BESKR.: ["Det man har brukt" — 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.

Chapter 4

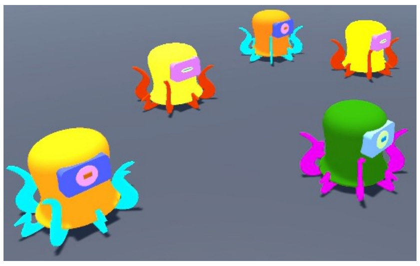
Implementation

This chapter gives an overview of the developed musical multi-robot system. The main goal of the implemented system is to allow for a multi-robot (musical) collective to interact with each other in order to achieve emergent and coordinating/co-operative behaviour—synchronization specifically in our case—with varying degrees 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. What is meant by *harmonic synchronization* will be expounded in Subsection 4.1.3. These goals firstly require of the agents/nodes the modelling of oscillators with their properties, like phase and frequency, as explained further in Subsection 4.1.1. 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 4.1.2.

4.1 The baseline: Achieving Decentralized Harmonic Synchronization in (Oscillators $\vee$ Musical Robot Collectives)

Envision that we have a multi-agent collective scenario consisting of musical robots modelled as oscillators, solely communicating through brief "fire"-like audio-signals—greatly inspired by K. Nymoen et al.'s approach for achieving *decentralized harmonic synchronization in mobile music systems* [1]. These agents are initially not 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 or after hearing each other's audio-signals. An illustration of this is given in Figure 4.1.

These aforementioned audio-signals to be expounded further in Subsection 4.1.2, also referred to as "fire"-signals, are transmitted whenever an agent's oscillator *peaks* (i.e. after its cycle or period is finished, having phase $\phi(t) = 1$)—or actually every second *peak*, due to the target system goal of *harmonic synchrony*. All agents have the ability to listen for such transmitted "fire"-signals from their neighbours, which they then will use as a trigger to adjust


(a) The agents firing asynchronously at first. Here, only the two Dr. Squiggles with red tentacles are firing simultaneously, but the rest are not.

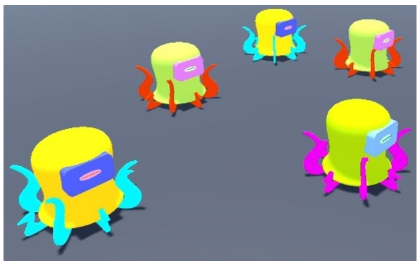

(b) Seconds later, after having listened to each other's fire-event signals and adapted themselves accordingly, the agents are here firing synchronously.

Figure 4.1: Decentralized Synchronization of phases achieved in a musical robot collective, consisting of M. J. Krzyzaniak and RITMO's Dr. Squiggles.

themselves according to some well-designed update-functions to be elaborated in Subsection 4.1.4.

4.1.1 The Node: the musical robot individual

BESKR.: [Om den enkelte noden/agenten med alle egenskaper den har osv. (som f.eks. en oscillator-komponent (jf. I.A., III.Intro., og 'Implementation' i Nymoens Firefly-paper))].

Notions like "agent", "node", "firefly", and "oscillator" will be used interchangeably throughout the thesis.

4.1.2 Robot communication: the "fire"-signal

No individual agent is able to adjust or modify the state of any other agent, and the only means of communication between the agents will be through these "fire"-signals.

- Short and impulsive audio sound/signal, representing a node's phase-climax and "fire"-/"flash"-event.
- Stigmergic co-ordination/communication.
- Facilitating PCO (pulse-coupled oscillators), different from phase-coupled oscillators with their differences.

GJØR: [Sjekk oppsummeringen av Nymoen-paperet i Essayet].

4.1.3 System target state: Harmonic Synchrony

- The goal and target state of the system

- All agents/nodes having integer-relation frequencies (between each other), so that given a fundamental frequency $\omega_{i,0}$ for the agent i with the lowest frequency in the collective — all other agents have "legal" harmonic frequencies; meaning, all nodes have frequencies in the mengde $\omega_{i,0} \cdot 2^{\mathbb{N}_0}$.

- (FOR Å DEMONSTRERE *Harmonic Synchrony*) LEGG INN ETT WAVEFORM-SPEKTROGRAM PLOTT AV EN STEMME ELLER EN LYD MED HARMONICS I SEG (F.EKS. FRA PRARIEDOGS-FYREN ELLER-NOE), OG SAMMENLIKN MED DE MUSIKALSKE NODENES FREKVENS-SPEKTROGRAM VED SIDEN AV.

4.1.4 Update/Adjustment functions: Phase- & Frequency-Adjustment

When it comes to the temporality and timing of when the updating 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.

4.1.4.1 Phase Adjustment

If we again 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. This was then in contrast to the case in Subsection 4.1.4.2 where heterogenous and randomly initialized oscillator-frequencies in the musical agents are implemented and utilized.

The goal state of the agents is for now to fire/flash more or less simultaneously, after having started firing/flashing at random initially. Note that this is a different goal from the final and ultimate goal of *harmonic synchrony*.

One relatively new approach to achieve this, firstly introduced by K. Nymoen et al. [1], is now presented:

Bi-Directional Phase Shifts

This Phase-adjustment, as in Figure 4.2, works very similarly to the Phase-adjustment performed in the "standard" ***Mirollo-Strogatz*** approach presented in the Background-chapter; The only difference being that now, nodes update their phase with the slightly more complex **phase update function** (4.1) when hearing a "fire"-event from one of the other musical nodes — allowing for both larger, but also smaller, updated phases:

$$P(\phi) = \phi - \alpha \cdot sin(2\pi\phi) \cdot |sin(2\pi\phi)| \tag{4.1}$$

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

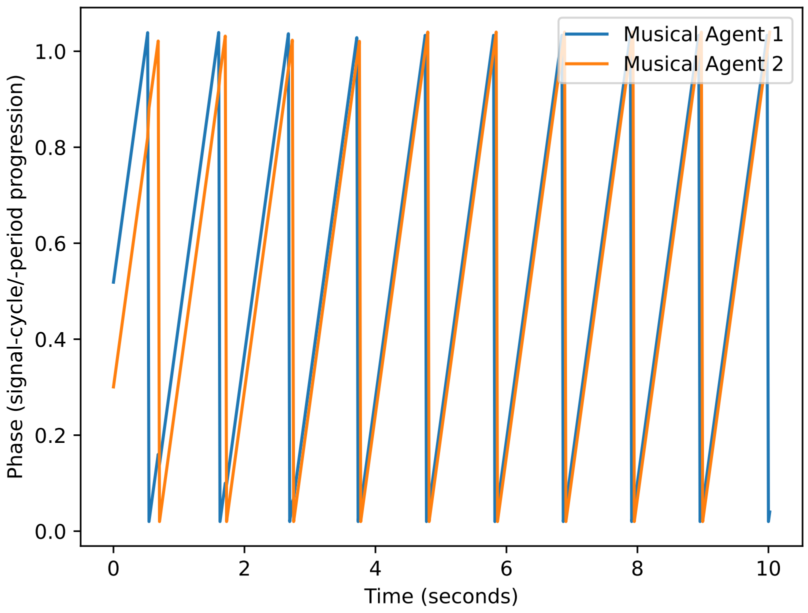


Figure 4.2: Bi-directional Phase-Adjustment with K. Nymoen et al.'s approach

being ***Bi-Directional***, or as the authors call it in the paper as using "*both excitatory and inhibitory phase couplings between oscillators*" [1].

The effects then of adjusting phases, upon hearing "fire"-events, according to this newest update-function (4.1) are that the nodes's phases now get decreased if ϕ is lower than 0.5, increased if ϕ is higher than 0.5, and neither—at least almost—if the phases are close to $0.5 = \frac{1}{2}$. This is due to the negative and positive sign of the sinewave-component in Equation (4.1), as well as the last attenuating factor in it of $|sin(2\pi\phi)| \approx |sin(2\pi\frac{1}{2})| = |sin(\pi)| = |0| = 0$.

4.1.4.2 Frequency Adjustment

GJØR: $\Big[$ Presenter hva Frequency-adjustment er, bygg opp fremgangsmåten på hvordan man oppnår Frequency-adjustment (som i Worklog'en, og Mattermost-chatten med Kyrre evt.) — med passende figurer og illustrasjoner $\Big]$.

Now over to the slightly more complex issue of synchronizing frequencies in the agents/oscillators. We introduce randomly initialized, non-constant, and heterogenous oscillator-frequencies in our musical agents — allowing in the music collective the playing of rhythmic patterns like doubles, fourths, eights (FRA NYMOEN: subdivisions of a measure, i.e. quarter notes and 8ths. "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)") etc. The agents are now then required to not only synchronize their phases to each other, but now also their initially different and random frequencies — so that frequencies are "legal" and *harmonically synchronized.*

This approach to Frequency Adjustment stands in contrast to previous approaches to synchronization in oscillators cite(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.

The goal state now will not be exactly equal frequencies and phases, but rather the goal state of *harmonic synchrony*, as expounded in Subsection 4.1.3.

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.

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

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{4.2}$$

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 a node should update its frequency after having received its nth "fire"-signal.

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

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 (4.3), plotted in Figure 4.3, 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.

$$\epsilon(\phi_i(t)) = sin^2(\pi\phi_i(t)) \tag{4.3}$$

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). 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 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. ??? **INKL.:** [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.] **?**

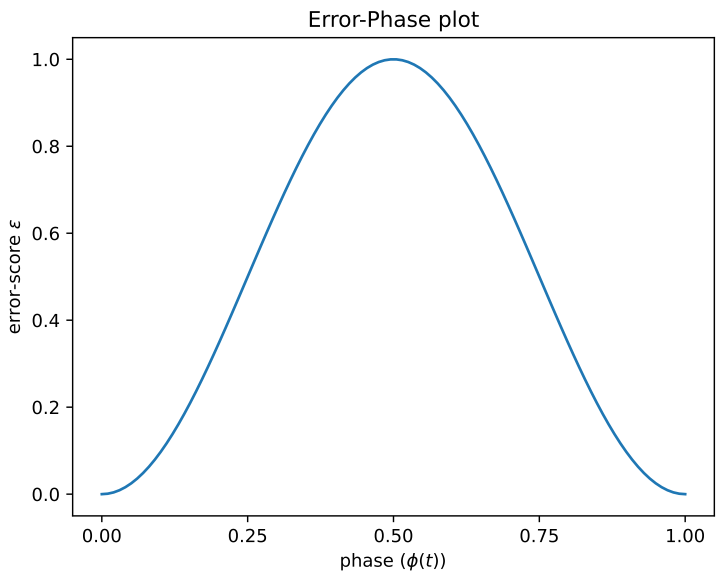


Figure 4.3: Error Measurement (4.3) plotted as a function of Phase

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

Step 2: The first Self-awareness component, s(n)
This aforementioned self-assessed synchrony-score, $s(n)$, is in fact simply the median of a list (was originally in K. Nymoen et al.'s approach a running median filter over the discrete error measurement function) 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):

$$errorBuffer = \{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}, \tag{4.4}$$

then leading to:

$$\begin{aligned} s(n) &= median(errorBuffer) \\ &= median(\{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}) \in [0,1], \end{aligned} \tag{4.5}$$

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.

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, we then have a way for each agent to assess themselves 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.

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.

$$\rho(\phi) = -sin(2\pi\phi(t)) \in [-1, 1] \tag{4.6}$$

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

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:

$$H(n) = \rho(n) \cdot s(n) \tag{4.7}$$

Here we then multiply a factor $\rho(n)$ representing how much, as well as in which direction, the node should adjust its frequency, together with a factor $s(n) \in [0, 1]$ of the adjusting node's self-assessed synch-score. To recall, the self-assessed synch-score $s(n)$ tells an adjusting node 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 node 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 node was at the time the node'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 the list of error-scores, *errorBuffer* in our case.

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.

All the calculated H-values are in my implementation 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.

The final step: The Frequency Update function

Putting it all together.

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

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{4.8}$$

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

$$F(n) = \beta \sum_{x=0}^{k-1} \frac{H(n-k)}{k}, \tag{4.9}$$

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 (4.9), is a weighted average of all the node's $H(n)$-values accumulated throughout the node's last cycle.

4.2 My new proposed algorithm: an additional Self-Awareness component

Chapter 5

Experiments and Results

BESKR.: [Skal følge opp *research questions*'a mine ved en diskusjon av til hvilken grad og på hvilke måter arbeidet har besvart dem].

GJØR: [Vurder å dele opp som Tønnes: Først 1) Evolusjonære/Simulator$^?$ eksperimenter og resultater, så 2) Fysiske eksperiment og resultater].

GJØR: [Legg inn performance-plot av initielt Simulator-eksperiment for synkroniseringstider for f.eks. Mirollo-Strogatz vs. Nymoen et al.'s fase-justering].

Chapter 6

Discussion/Conclusions

GJØR: [Se på Tønnes sin masteroppgave for inspirasjon].
GJØR: [(Hvis jeg heller vil bruke ’Conclusions’ som tittel og dra inspirasjon fra Jim) Eller se på ’how to write a master thesis’ av Jim om ’Conclusions’].

Bibliography

[1] Kristian Nymoen et al. “Decentralized Harmonic Synchronization in Mobile Music Systems”. In: *Grant agreement no. 257906 (EPiCS) from EU FP7* (2014).