Computational Self-Awareness in Musical Robotic Systems

Endowing musical robots with self-awareness — an experimental and exploratory study

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

Endowing musical robots with self-awareness — an experimental and exploratory study

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.

Soli Deo Gloria

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

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. As we can see in various Music Technology Systems, this endowing can also give rise to interesting cooperative and coordinating behaviour.

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).

1.2 Goal of the thesis

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

BESKR.: [Spesifikke mål (goals/aims) med master-oppgaven/-prosjektet. Hva jeg vil vise/demonstrere til folk (leserne f.eks.) om self-awareness (SA) og hvordan (detaljene av beskrevet i **Implementation**) — noe jeg håper å enten bekrefte eller avkrefte med eksperimenter og resultater i **Experiments and results**].

The specific aim of this thesis is to explore and investigate the effects of endowing robot systems with increased self-awareness abilities, and thus leads to the following research questions:

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.

1.4 Scope and delimitations

BESKR.: [Hva oppgaven forsøker å oppnå eller besvare, og hva den ikke forsøker å oppnå eller besvare].

1.5 Contributions

BESKR.: [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). Fra Mats: HUSK Å SELG!].

This thesis work has led to some main novel contributions. The main contribution is that a novel synchronization-simulator has been created and developed in Unity, where various and customly made synchronization-methods can be implemented in a robot collective (either homogenously but also heterogenously in each robot); as well as that users of the simulator can both see and hear the robots's interactions and entrainment towards synchrony (including whether they manage to finally achieve synchrony, and in that case how long it takes them to get there). This novel framework opens, for everyone interested in it, for the possibility of experimenting with creative and novel synchronization-methods, in order to qualitatively and quantitatively assess their efficiency at achieving synchrony. A reason for this is due to the seamless scalability and dynamism, both in terms of robot-collective size, heterogenous or homogenous agents, as well as how a new update-function (for both phase-updating and frequency-updating) can so easily be added in and coded with a couple of lines.

The developed synchronization simulator in Unity, which is open to the world and those interested in it, serves as a testbed for seeing how changes in robots

synchronizing affects synchronization performance; if e.g. someone thinks they have an idea for making the robots more clever, they can easily download the Unity simulator, implement their approach, and see analytical and graphical (as well as audible) results.

Chapter 2

Background

2.1 Taking inspiration from psychology

Self awareness (SA) in psychology:

The notion and idea of *self-awareness* has been of great interest to humans throughout history. And despite its sometimes abstract and elusive nature, many thinkers (like e.g. psychologists and philosophers) have previously tried to give an explanation and description of this phenomenon or capability, primarily as observed in individual humans and animals. One has also tried to extend these descriptions and developed concepts to explain collectives consisting of individuals. A good survey of such descriptions and explanations regarding Self-Awareness in Psychology is given in some recent books on Self-Aware Computing Systems [8, 9] by P. R. Lewis et al, and will be used as a basis for this section and essay.

The notion of self-awareness might give associations to other psychological terms like perception and consciousness, and we will look at some relationships these have with the main notion here of self-awareness, in the psychological and (briefly) philosophical literature.

What exactly is self-awareness? Is it *awareness* of a *self*? If so, this leads to the question: what constitutes a *self*? And also – what does really *awareness* mean?

2.1.1 Awareness

Before Lewis et al. [9] inclusively and succinctly introduces the concepts and interpretations relating to Self-Awareness, as they pertain to the psychological literature especially, they start off by giving the definition of *awareness* by The *Oxford English Dictionary*, which is "knowledge or perception of a situation or fact." Hence (by simply adding the prefix *self*), "situational or factual knowledge/perception about a *self*" might seem to be a reasonable first intuition of the notion of Self-Awareness. We will now discuss further what might constitute a so-called *self*.

2.1.2 The *Self*

Prominent western philosophers, like David Hume and Immanuel Kant, had some differing views on what constitutes a *self* – although both might seem reasonable and relevant.

As Lewis et al. describes it [9], Hume is known for postulating that all human knowledge is induced from experience. This then means for Hume that the *self* is no concrete and physical entity in the world, but rather the bundle of experiences or perceptions unique to an individual.

On the other hand, for Kant, the scope of the self is significantly extended – as he argues for the self being an actual entity being the subject of experiences, which is common through space and time. This subject or *self* then combines information from experiences with concepts in the mind, as well as with the imagination – as well as acting upon this combined knowledge.

2.1.3 Self-Aware Individuals

Given a Humean view of the self (as described above), one might then consider self-awareness to consist of an individual's knowledge of its experiences. Whereas the Kantian view of the self yields a self-awareness consisting of a distinct and physical individual subject to experiences, combining information from these experiences, concepts, and the imagination.

Early appearing terms in the psychology literature, pertaining to Self-Awareness and notions of self – being reminiscent of these two differing views by Kant and Hume – are found in the distinction between the *objective/explicit self*, and the *subjective/implicit self*, firstly elucidated by W. James in 1890 [**james_explicit_implicit_self**], and further expounded in detail by Duval [**duval111**]. Here, the *implicit* self, often referred to as the self-as-subject, is the "me"-self, subject to experiences. The self-awareness in this "me"-self then, according to Duval, is described as "*a state of consciousness in which attention is focused on events external to the individual's consciousness, personal history, or body*" [**duval111**]. The *explicit* self however, often said to be the "I"-self, describes a self-as-object which can be discerned, observed, and reasoned about in the world. This "I"-self then, according to Duval, has an objective self-awareness described as being "*focused exclusively upon the self and consequently the individual attends to his conscious state, his personal history, his body, or any other personal aspects of himself*" [**duval111**].

One well-known "self-awareness-test", having been used to deem whether someone or something is self-aware or not, is the so-called *mirror test*. This consists of observing an individual's capability (or lack thereof) of noticing or self-recognizing, in a mirror, a discreet change in their own appearance (that was caused by someone else typically, while they were not paying attention e.g.). The presence of such a capability is argued (e.g. by Asendorpf et al. [**asendorpf**]) to demonstrate the presence of a *secondary representation* (a constructed conceptual model if you will) of oneself and the world (the primary representation being the real world).

Given that the ability to construct such a conceptual model of oneself is required in explicit self-awareness, one could wonder whether passing this "mirror test" would then be satisfactory for someone or something (like a system) to be self-aware. Haikonen [**haikonen**] however demonstrated how easy it was for a

not-too-sophisticated computing system to pass this test – and hence weakened the validity of this "mirror test" to definitively conclude whether someone or something was indeed self-aware or not.

As should be more and more apparent, there has been – and still is – a lot of ongoing discussions about what should be considered self-awareness, and what should not – as well as about all the different variants observed (by e.g. Morin [**morin**]).

Morin defined self-awareness as different from, but building upon the previously mentioned consciousness (where this in some cases are deemed more "primitive" aspects of self-awareness), in that "self awareness is the capacity to become the object of one's own attention." However, he doesn't deny the subjective aspect of the self-aware individual, in that he believes a self-aware organism to be one that "becomes aware that it is awake and actually experiencing specific mental events."

The importance of including such a *subjective* (perhaps more underlying and primitive) level of self-awareness, lies in the realization that self-aware agents (being uniquely situated in the world, collecting unique data from their unique sensing-apparatus) **is** subjective in nature. As such, since possibly heterogenous agents might collect widely different data (due to their situation and sensing capabilities) from the same exact phenomenon, it is crucial that we know from which *subject* the data we have originates from.

Perhaps or perhaps not due to this, others would object to Morin's definition, and actually include this subjective and implicit (or in other words "perceptual (or pre-conceptual)", as Lewis et al. [9] put it) experience in their definition of self-awareness – leading to self-awareness and consciousness being overlapping concepts. This would also lead more complex and "full-stack" Self-Aware systems to being extensions of this simpler and more primitive level of Self-Awareness.

2.1.4 Public and Private Self-Awareness

The distinction, as discussed above, between the implicit/subjective and the explicit/objective self and corresponding self-awareness has given rise to many further developments in this notion, as is presented by Lewis et al. [9]. However, these further developments separate this seeming distinction of *private self-awareness* (previously described as e.g. personal, historical, and bodily knowledge regarding the self-as-object) and *public self-awareness* (previously described as awake, subjective, and unique experiences external to the self) a bit differently.

Rather than the above notions of public and private self-awareness, **private self-awareness** instead here refers to the obtaining of "knowledge of internal phenomena, typically externally unobservable and accessible only to the individual" [9]. On the other hand, **public self-awareness** now concerns the gathering of knowledge based on phenomena external to the self. If we choose to look at public self-awareness in the implicit/subjective form, we have a "minimally" self-aware individual who is merely able to perceive (subjectively) the external environment in which it is situated. Conversely, we could also have a publicly self-aware individual in the explicit/objective form – which would be more concerned with public appearance, and how the individual is perceived in its in

environment (e.g. by other individuals, in social settings). These definitions of private and public self-awareness will be used from now on.

2.1.5 Levels of Self-Awareness

There is large consensus about, due to the many degrees or variations to which an individual can be self-aware, that self-awareness is not *binary* or singular – but rather on a spectrum. As Lewis et al. highlights ([8, 9]), varieties of an individual's self-awareness capabilities are often associated or characterized with one or more levels of self-awareness. As such, several have attempted at defining such self-awareness levels.

One such definition, being comparatively broad and wide (to other similar treatments), containing 5 increasingly complex self-awareness levels, is given by Neisser [**neisser284**]:

1. **Ecological self**

   The ecological self is the most minimal form of self-awareness. It permits sufficient knowledge only for basic stimulus-response behaviour, as the individual has a basic awareness of stimuli. The ecological self can be thought of as the minimum requirement for the individual to not be unconscious.

2. **Interpersonal self**

   The interpersonal self enables the individual to possess a simple awareness of its external interactions, permitting limited adaptation to others in the performance of tasks.

3. **Extended self**

   The extended self extends the interpersonal self to permit reflection of interactions over time. The individual is aware of the existence of past and future interactions.

4. **Private self**

   The private self allows the individual to process more advanced information concerning itself, such as thoughts, feelings and intentions.

5. **Conceptual self**

   The conceptual self, or self-concept, is the most advanced form of self-awareness, representing that the individual is capable of constructing and reasoning about an abstract representation of itself.

This final level, the **Conceptual self**, is very much connected to the much-discussed term of *meta-self-awareness*. Meta-self-awareness refers to the awareness of someone that they themselves are self-aware. This includes reasoning about its own self-awareness, and is argued to be necessary (at least in humans) in order to adapt to changes, and to direct ones attention where it is needed – and that without it, thoughtlessness would ensue.

We will see later on in section **4** how these levels come into play in a fundamental basis and a proposed reference framework, for describing Computational Self-Awareness.

2.1.6 Self-Aware Collectives

In her comparison of three distributed and collective biological systems (the brain, the immune system, and ant colonies), Mitchell [**mitchell**] elucidates how – not only individuals, as has been the case until now, but – also collectives (consisting of individuals) can exhibit apparent global self-awareness properties/capabilities. As she goes on to explain and make her case, she further elucidates how these apparent self-awareness properties are only seen globally – but not in individuals (which in this case are neurons, ants, or lymphocytes - also known as white blood cells). The explanation for this, she proposes, is due to the distributed and statistical nature of this apparent self-awareness. Through the many individual interactions in the collective – self-awareness gets built up *bottom-up*, and then gets fed back down to the lower-level components in a *top-down*-fashion.

Given that this apparent self-awareness then was not present in the collective's individual components, but in the global collective, as a result of the many interactions of the lower-level components – we might say that the collectives exhibit ***emergent*** self-awareness.

Hence, we can both analyse and describe Self-Awareness capabilities in individuals, as well as collectives of individuals.

2.1.7 Self-Expression

Lewis et al. [9] argues that self-awareness properties alone (i.e. the ability to continuously acquire, update, and synthesise the self's models and knowledge about itself and its experiences) is of limited value, unless it is accompanied by associated behaviour. As Chandra et al. [2] argue, later on in the same book – if the self only takes data in, but does not use that data to do anything, then it is essentially a "data-sink". Hence, the accompanying notion of *self-expression* is introduced as "behaviour based on self-awareness" [9]. This behaviour could be highly dynamic (e.g. enabling self-adaptation, self-reconfiguration, self-explanation), as well as constituting a more "standard" system behaviour, where self-awareness is not considered at all.

2.2 Self-Awareness in already-existing Computing Systems

2.2.1 SA-related Computing Systems

Run-time adaptation in Computing and Engineering systems is not a new thing, and this has been of interest across a range of research communities for quite some time [2]. The reason for this becomes apparent when one considers the problem of tackle unforeseen (at least at design-time) operational changes that can occur during run-time (like e.g. faults).

An agent – in the context of computing systems – can be said to be a computing system situated in an environment (which it can perceive), having some goals, and possible actions it can perform (through actuators) to reach its goals. It can also be informed by some knowledge, and/or directly the signals it gets from its sensors.

Especially within the engineering of **Self-adaptive** Systems and Intelligent Agents, such agents are often designed with properties which can be called Self-Awareness and -Expression properties – even though these specific terms are not used.

Notable examples in accordance with the autonomic computing paradigm ([2]) comprise for example ODA (Observe-Decide-Act) loops, originally stemming from the OODA (Observe-Orient-Decide-Act) loop in the military domain. As the acronym suggests, systems implementing the ODA-loop continually Observe (their environment and potentially themselves through sensors), Decide (planned future actions based on current or future states), as well as Act (on one of those previously planned actions). A schematic of such an ODA-loop can be seen in Figure 2.1.

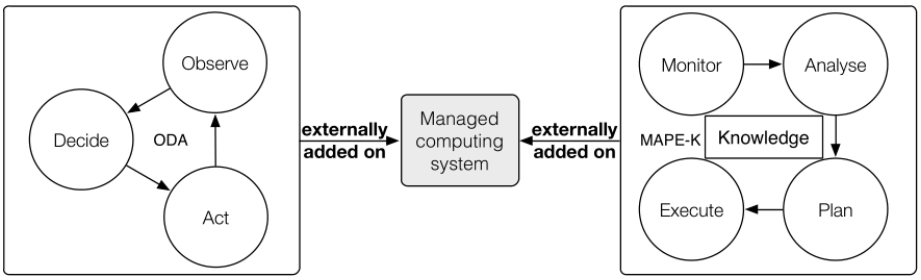


Figure 2.1: Figure of the Self-adaptive Systems ODA (left) and MAPE-K (right). We also see the externality of the attempts at implementing meta-self-awareness in these systems, in that they simply *add on* an external Managed computing system (which is monitoring the rest of the system). This latter point will be revisited later[?]. Figure reused from Chandra et al. [2].

Another notable Self-adaptive System we can see in Figure 2.1, is the MAPE-K architecture (an acronym for Monitor, Analyse, Plan, Execute, and Knowledge). The great and groundbreaking benefits of implementing e.g. ODA-loops and the MAPE-K architecture, performing online observation and knowledge-gathering (hence having endowed their systems with some sort of self-knowledge/self-awareness) allowing for online self-adaptation – have been greatly reaped. However, these architectures have not explicitly considered the psychological origins of self-awareness, and lack some benefits compared to the introduced frameworks and architecture in section **4**.

2.2.2 Explicit SA in Computing Systems

The extensive interest of studying self-awareness in natural fields like psychology, sociology, and philosophy might not come as a surprise, as self-awareness primarily is found and exhibited in humans and animals. However, there has recently been an uptake in interest for explicitly studying self-awareness in computing systems.

Recently, (often collaborative) efforts within the Computer Science & Engineering community [7, 8, 9] have lead to a greater systematic understanding of self-awareness, and how it can be used and evaluated within Computing Systems.

Some of these fields, now explicitly interacting with the notion of self-awareness – as they were presented in overviews given by P. R. Lewis et al. [8] and S. Kounev et al. [7] – include "*autonomic computing, machine learning and artificial intelligence, multi-agent systems, self-organizing and self-adaptive systems, situation-and context-aware systems, reflective computing, model-predictive control, as well as work from the models@run-time community*" [7]. But also, fields like Robotics, complex IT systems, computer engineering, and reflective architecturing [8].

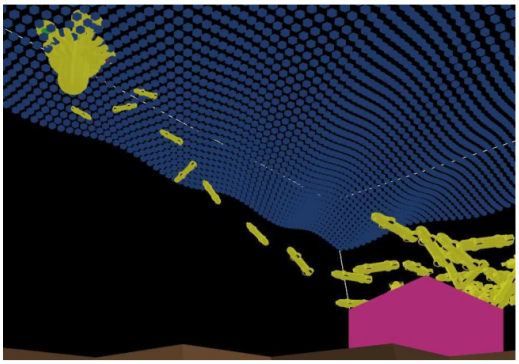
Some quite notable research projects and initiatives within computer science and engineering, explicitly engaged with the notion of self-awareness in computing has been performed (like the SEEC project at MIT and University of Chicago, the ASCENS and EAssets/pics FP7 EU Projects, as well as the SEAMS Dagstuhl Seminars and workshop series) [7].

2.2.2.1 Specific Examples

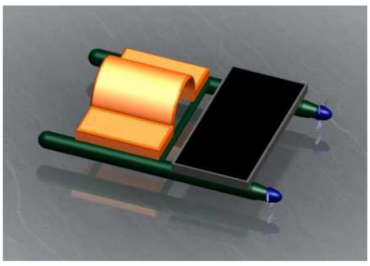
Cognitive Radios A – perhaps simpler or minimal – specific instance of a computing system where Self-awareness have been interpreted and applied, is the so-called *cognitive radio*. The devices in this system control their own capabilities and monitor other devices' capabilities through communication with these other devices. This "*enables them to improve the efficiency of communication by negotiating changes in parameter settings*" [8]

CoCoRo - The Self-aware Underwater Swarm More complex self-aware computing systems have also been explored. As demonstrated by Mitchell [**mitchell**], also T. Schmickl et al. considers self awareness in a collective system – an underwater swarm robotic system [14] in this case (see Figure 2.2). This robot collective shall in this scenario search the sea-bottom for specific objects of interest (like toxic waste, but also sunken objects e.g.). Here, the collective/swarm is supposed to e.g. be self-aware of its own state (of whether it still is searching, or if it has found a target at the bottom of the sea) – in order to either stay together searching, or communicate to a second (surface level) swarm that an object of interest is found. Like in Mitchell's case, these autonomous underwater vehicles (AUVs) does have emergent self-awareness properties that the individuals themselves do not possess. Therefore, as the authors say in [14], "*Finding such targets will only be efficient if the AUVs work together.*"

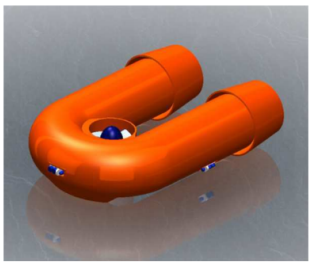
Self-Aware and Self-Expressive Camera Networks Lukas Esterle et al. [**esterle_camera**] has demonstrated how the fundamental concepts and building blocks of Computational Self-Awareness and Self-Expression can handle resource-critical- as well as communication- and utility-tradeoffs, maintaining network topologies, performing distributed object detection, performing tracking handover, as well as achieving more Pareto-efficient outcomes. They also believe these capabilities are not exclusive to camera networks – but are in fact confident these technologies (of computational SA and SE) can enable "*other types of systems and networks to meet a multitude of requirements with respect to functionality, flexibility, performance, resource usage, costs, reliability, and safety*" [**esterle_camera**].



(a) Swarm searching the seabed



(b) Basestation



(c) AUV

Figure 2.2: CoCoRo—the self aware underwater robotic swarm consisting of AUVs alternating between completing missions at the seabed and returning to the surface level basestation. Figure reused from T. Schmickl et al.'s paper [14].

2.3 Designing and Describing Self-Aware Computing Systems

Psychology has served as the basis & main source of inspiration. Here we will look at a translation of Self-Awareness, from Psychology to Computing. We will hence look at an introduction of the new notion of *Computational Self-Awareness*.

Translating concepts from the Psychological field into the Computational field gives rise to rich sources of inspirations (for engineers and researchers who might not have thought psychologically about designing and describing computing systems), as well as hopefully clear, structured, principled conceptual tools to in a consistent manner describe and design so-called *Self-Aware Computing Systems*.

2.3.1 Basis and Fundament

Lewis et al. [9] has introduced a basis and fundament for Computational Self-awareness and Self-expression, that is founded in the psychological literature on Self-Awareness (as presented in section **2**).

More specifically, they base their new notion of computational self-awareness on computationally translated versions of the key self-awareness concepts dis-

cussed in sub-sections **2.4**, **2.5**, and **2.6**, as well as **2.7**.

First off, public and private self-awareness now considers computational systems, instead of human selves – but these definitions are still the same (only that now the *self* is the *computational system*).

Secondly, Neisser's five levels of Self-awareness is now translated into five corresponding levels, as shown in **??** below.

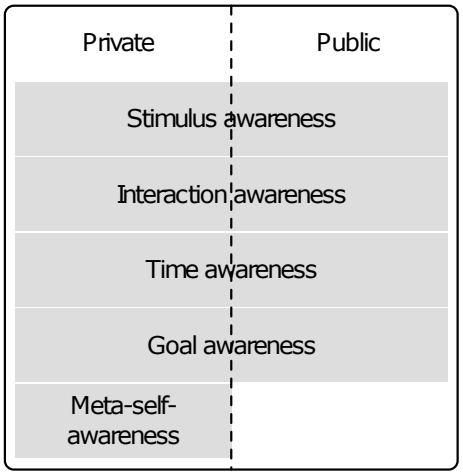


Figure 2.3: Translated levels of self awareness from psychology to computer engineering and robotics.

Thirdly, computational self-awareness can both be considered in individual agents/computing systems, or in arbitrary agent-collectives (consisting of several sub-collectives) – as seen in 2.4 below.

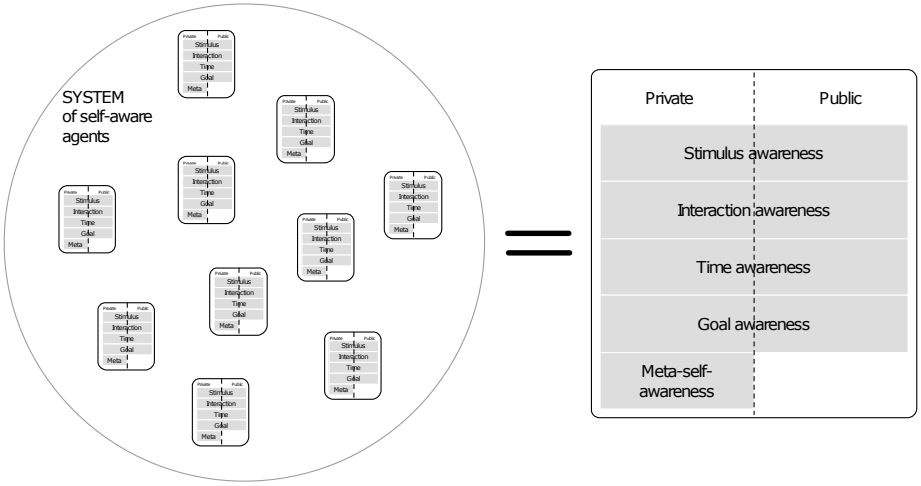


Figure 2.4: Collective self awareness, as entities can be seen as consisting of many sub-entities.

Lastly, *Computational Self-Expression* is similarly defined as (only now for a computing system as the *self*) **behaviour based on *computational* self-**

awareness.

Kounev & Lewis et al. also gives a definition of Self-aware Computing [7]:

Definition 1.1 Self-aware computing systems are computing systems that:

1. *learn models* capturing *knowledge* about themselves and their environment (such as their structure, design, state, possible actions, and runtime behavior) on an ongoing basis and

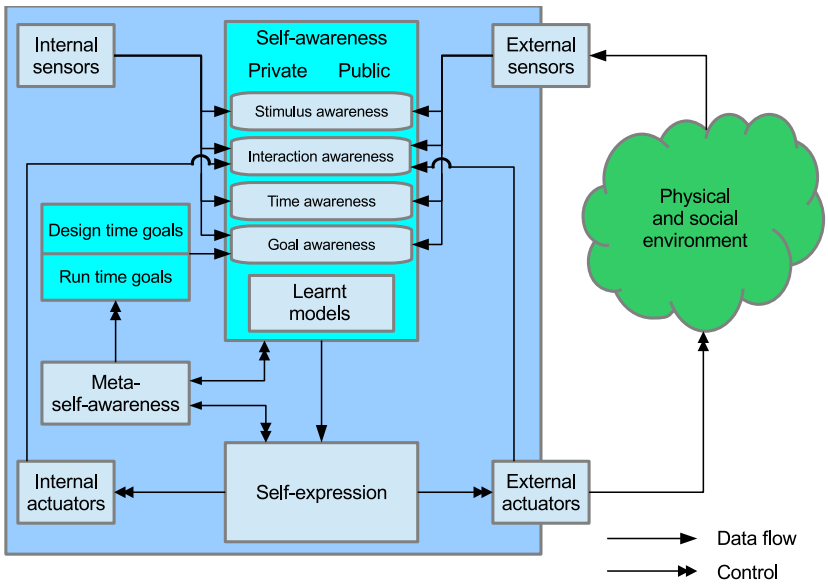
2. *reason* using the models (e.g., predict, analyze, consider, and plan) enabling them to *act* based on their knowledge and reasoning (e.g., explore, explain, report, suggest, self-adapt, or impact their environment)

– in accordance with *higher-level goals*, which may also be subject to change.

2.3.2 Reference Architecture

Chandra et al. introduces their own new Reference Architecture for designing and describing Computational Self-Awareness and Self-Expression [2]. This abstract and conceptual framework has several advantages. Firstly, it provides a detailed and fine-grained separation of different self-awareness and self-expression processes – and hence makes design- and engineering-questions easier and more methodical. Secondly, being a common language, it paves the way for identifying common architectural patterns used to design Comp. SA- & SE-systems. Compared to other Self-Adaptive systems, it also includes the notions of Self-Awareness from the beginning (as the psychology literature suggests SA capabilities is more innate than simply an add-on). And due to the general and implementation-agnostic framework, it can be used on a wide variety of agents systems, or agent-collectives systems.

A schematic of the reference architecture is given below:



This is expounded in detail in [2].

2.3.3 Conceptual Framework

Lewis et al. [8] introduces an inclusive and extensive Conceptual Framework, built and based on the psychological foundation and basis *(as well as expanding on the Reference Architecture)*, as introduced earlier in this section.

This Conceptual Framework is based on the following three high-level concepts/tenets:

1. **Levels** of self-awareness
    - **Pre-Reflective self-awareness** – ability to subjectively and uniquely perceive and experience phenomena in the physical-/social-environment as well as within itself. A pre-requisite for all other and more advanced forms of computational self-awareness, also implicitly introducing the notion of subjectivity.
    - **Reflective self-awareness** – modelling and conceptualizing various knowledge-/awareness-aspects about an *object*, where the object is what is modelled, or the sensory observation of it at least.
    - **Meta-Reflective self-awareness** – **Reflective self-awareness** processes about its own self-awareness processes. Reasoning and modelling of the oneself's own (as well as potentially other systems') self-awareness properties and processes. Typically its *objects* are 1) the self-awareness (modelling) processes themselves, and 2) the output of these processes, i.e. the models themselves. And often the combination (e.g. the memory usage of the learning of an AI model, versus the accuracy of the model – hence potentially calling for a change in architecture or algorithm-choice in a self-awareness process) are useful.
2. **Aspects** of Reflective and Meta-reflective Self-awareness
    - All the different aspects and facets of models being learnt and reflected with (like temporal aspects, interactive/causing, identity and state aspects, behavior awareness, appearance awareness, goal awareness, belief awareness capturing uncertainty and trust-levels, expectation awareness combining belief- and time-awareness).
    - Not a comprehensive and large list of requirements a system has to check off in order to be considered Self-Aware (due to specific context or system design situations calling for different needs) – but rather a good starting point for considering the aspects captured in reflective and meta-reflective self-awareness.
3. **Domain** of Self-awareness, being the combination of the two;
    - Span (subject(s) of reflective self-awareness, comprising all entities that contribute to the formation of self-awareness)
    - Scope (object(s) of self-awareness, comprising all the entities observed by the subject's self-awareness). E.g., in the case of robotics, the number or amount of neighbouring robots they are aware of and communicate with.

From [8]:

"*Using the intersection of all three tenets, we may produce descriptions of the specific self-awareness of a system. For example, we might construct a sentence as follows: Peter (span) is aware of Ada's (scope) goal (aspect) to reduce the power usage (object). Further, Ada (span) is aware of her own reasoning (meta-self-awareness) about what to do (act) about it.*"

2.3.4 Challenges & Limitations

Physical constraints like computational processing power, as well as time, limited attention e.g. are all significant constraints which designers of Self-Aware and Self-Expressive Computing Systems have to keep in mind.

2.4 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].

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. Such bio-inspired algorithms have been some of the most widely used optimization algorithms throughout history, with e.g. the original paper by M. Dorigo et al. for the ant colony optimization algorithm [3], as of May 2022, having 14 743 citations on Google Scholar.

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) [4], to ... Such phenomena have been subject to considerable study and modelling—and have in fact created entire scientific fields in which principles from various science- & engineering-dicipline 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*[1] 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 2.5.

[1] *Nature*, `https://www.nature.com/subjects/biomimetics` (accessed 2022.05.17)

Figure 2.5: Synchronous fireflies at Congaree National Park, United States. Photo[2]: @columbiasc & @_flashnick.

This phenomenon has inspired scientists like Mirollo & Strogatz [10] and in later time Kristian Nymoen et al. [12], 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 [10, 12] 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 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.

2.5 Signals and their responses

GJØR: [Forklare hva et signal eller et (lyd-)signal/waveform egentlig er (ift. harmonics/harmonic wave/overtones (og deres typiske integer-relationship frekvenser), og hvis tid timbre med varierende frekvenser (jf. Jay Eller hva han Andriy Burkov eller hva han het anbefalte), amplitude/gain, envelope/attack-decay)].

Signals, in various forms, are omnipresent in our world, whether we notice it or not. They are used for guidance: e.g. traffic-lights send visual light-signals to drivers to ensure traffic-flow and collision-avoidance; sirens and alarms fire away with the loudest of audio-signals (sounds) so that its listeners will get out of harms way; our nerves send pain-signals through our nervous-system if we touch something burning hot to ensure we do not damage our hand severely. If you have ever tried to make a scary sound to scare away an animal—like a cat or crow—they might in fact flee from you if they think your audible signal was scary enough for them.

A signal can also be called a waveform.

[2] `https://avltoday.6amcity.com/synchronous-fireflies-congaree-national-park/` (accessed 2022.05.17)

2.5.1 Amplitude

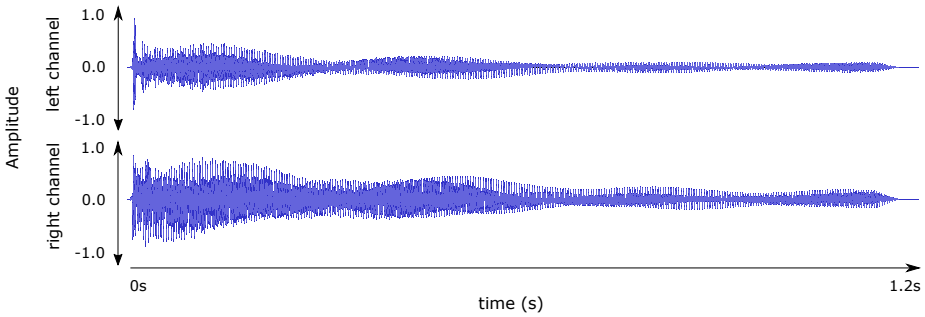


Figure 2.6: An audible stereo- (hence the two channels, one for each ear) waveform, being a digitally created harp-sound (as further described in 5.1.3). The more positive or negative the amplitude is, the louder the audio-signal is.

2.5.2 Frequency

2.5.2.1 Spectra / Spectrograms

2.6 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. Nevn fordelene, spesielt for roboter, med å synkronisere seg basert på diskret tid [Sandsbots-paperet Kyrre sendte deg (iaff. slutt-tabellen), SandBots-arkitekturen som åpner for lave oppdaterings-rater], sammenliknet med kontinuerlig tid].

BESKR.: [Teori og nyere cutting-edge/state-of-the-art metoder for fase-/frekvens-synkronisering i oscillatorer og liknende. Se Kristian Nymoens referanser [12] og artikler funnet i denne veinen].

Oscillators have been used to implement and model a plethora of systems, also biological ones, ranging from designing the locomotion-patterns of swimming robot-amphibia through central pattern generators [5], ..., and as we have already established, modelling synchronously flashing or firing fireflies.

2.6.1 Phase and frequency

Much of the terminology from [12] 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:

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

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

2.6.2 Frequency-adjustment

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

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

BESKR.: [(If it exists) A "simpler" frequency synchronization approach, to solve the phase and frequency (ϕ & ω) problem with, without any self awareness components].

2.7 Computational Self-Awareness in music technology systems and robots

As particularly relevant to the thesis project, endowing Music Technology Systems with Self-Awareness is of great interest. There have been several such approaches, particularly within Interactive and Active Music Systems. One example includes:

2.7.1 SoloJam: Achieving de-centralized participation (by co-ordinating and co-operating agents) in a coherent and interactive music system

Historically, the creation and production together with the listening and perception of music has been (and still is) relatively "one-way'ed"; in that the musicians/artists produce the music, and the fans/listeners consumes the music. A growing interest in interactive music (like Guitar Hero etc.) has been

seen the last decades. The authors of the music system SoloJam [1] want participants with little or no musical experience and training to be able to play independently and decentralized (from say an expert) music as a whole, by each playing solos made from their own devices. Here each of their solos will be slight variations of the previous participant's solo. See illustration in Figure 2.7.

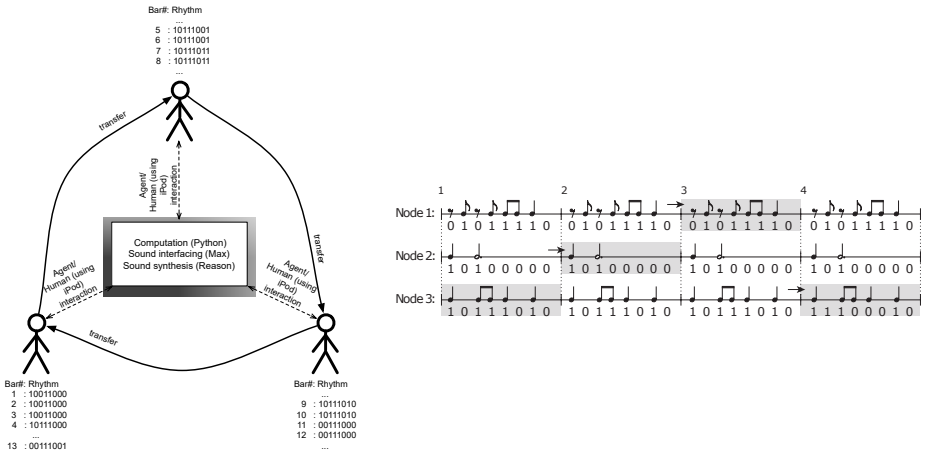


Figure 2.7: Illustrations: two individual SoloJam example scenarios illustrated differently.

The solution to how this decentralized, coherent and interactive music system consist of elements inspired by the Economical Sciences. Namely:

1. Auctions (specifically the Vickrey auction)
2. Utility (profitability/deservedness measure)

The function for utility is given by the expression:

$$u_i = \frac{c}{(1 + aD_l)(1 + bT_l)} \tag{2.3}$$

It is essential with a properly designed utility function for the system to behave coherently e.g. This utility-function is inversely proportional with *a)* differences between bidder and auctioneer's/leader's rhythm patterns D_l, and *b)* the time/#bars the leader has played its rhythm pattern T_l - as well as constrained by and applied to a couple of other clauses (to ensure some wanted/avoid some unwanted effects).

"Shaking"-/"moving"-patterns (as registered/sensed by the devices' inertial units) are translated into rhythm patterns - which then are calculated a **utility score** (by the function as mentioned above and outlined in more detail in the paper) for, used as bid in an auction.

The winner of the auction will (when considering knowledge of the leader-rhythm-pattern, as we do when a = 1.0 instead of 0.0) be those rhythm patterns which are the closest rhythm-patterns (i.e. not completely equal) to the leader's rhythm-pattern (in terms of the Hammings-distance) - assuming e.g. that the

participant didn't just play before another participant, nor that it had played its rhythm-pattern for a very long time. In the paper, we then notice how the authors study how **varying degrees of self-awareness** (a=1.0 versus a=0.0) affect the overall system dynamics – then in the form of musical coherence, as well as decentralized circulation.

2.7.2 Musical robots: the Dr. Squiggles

M. J. Krzyzaniak and RITMO's musical robots, the Dr. Squiggles[3], have been used in several music technology systems and projects previously.

2.7.3 SoloJam Island

Corresponding 3D-models of the Dr. Squiggle robots, for simulation, were designed by Pierre Potel[4].

Combining the two above.

[3] `https://www.uio.no/ritmo/english/news-and-events/news/2021/drsquiggles.html` (accessed 2022.05.17)

[4] *SoloJam-Island*, `https://github.com/67K-You/SoloJam-Island` (accessed 2022.05.17)

Chapter 3

Baseline

BESKR.: [Mellomkapittel om allerede-eksisterende approaches og teori jeg brukte som starting point og som jeg verifiserer og sammenlikner med senere].

Kristian Nymoen et al. showed in their paper titled *Decentralized Harmonic Synchronization in Mobile Music Systems* [12] how one can achieve synchronization (*harmonic* at that, cf. 3.1) of pulse coupled oscillators's initially unsynchronized phases and frequencies. This is done by endowing oscillators with self awareness capabilities amongst other measures. The platform Nymoen et al. achieved this on is shown in Figure 3.1, as each firefly on their iOS devices initially start flashing unsynchronized and randomly, but then eventually manage to start firing synchronously[1].

Key words: **Active & interactive music, decentralized, synchronization, pulse-coupled oscillators, self-awareness, emergence & stigmergy**

In this Active Music System, as shown in Figure 3.1, decentralized harmonic synchronization (implying integer-ratio oscillator-firings) of a dynamic and arbitrary number of musical nodes (pulse-coupled oscillators) – involving indirect coordination through stigmergy, as well as self-awareness through knowledge regarding itself internally and in publicly in the context of other nodes – is achieved.

Figure 3.1: Decentralized oscillators modelled as fireflies achieving harmonic synchrony on iOS devices, after starting flashing unsynchronously. Image: K. Nymoen et al. [12].

[1] See video at `https://vimeo.com/67205605` (accessed 2022.05.17)

Communication:

- solely through audio
- indirectly in the environment $\implies$ facilitating stigmergic coordination

Synchronization:

- **Synchronizing** the **phase** (the cycles, interval-/period-starts). Utilizing bidirectional phase shifts (dependant on being in first or second half of phase cycle, i.e. $\phi < 0.5$ or $\phi > 0.5$), either subtracting from the phase, or adding to it (based on the phase update function, $P(\phi)$).
- **Synchronizing** the **frequency** (the speed the oscillator cycles through its cycle/period) through:
  - The implemented **self-awareness** in the **self-assessment of its own synchrony** (calculated by the discrete error function and a running median filter over this, $s(n)$, multiplied by a discrete amplitude- and sign- frequency-modification-function $\rho(n)$ – yielding $\rho(n)s(n) = H(n)$).
  - Werner-Allen's *reachback firefly algorithm (RFA)*, involving collecting received fire events and updating frequency only every second round, as opposed to always updating frequency at once a fire event is received.

As Nymoen designed update functions for synchronizing both phases and frequencies, not only were their firefly-inspired oscillators endowed with self awareness capabilities; they were also designed to adjust their phases and frequencies with (close to) 0 change half-way through the oscillator cycle. These (close to) 0 changes half-way through oscillators's cycles enable individual musical agents to play with different measures (e.g. half or 1/4th as fast) compared to other individuals—being characteristic for musical interaction with several musical individuals—as well as enabling the possibility of achieving the target state of harmonic synchrony (cf. 3.1). Their contribution also differs from previous approaches in that Nymoen et al.'s fireflies only need to transmit "fire" signals every other phase climax, not every single phase climax; in other words, synchronization is achieved despite less communication.

First, the relatively new system target state of *harmonic synchrony* will be expounded and explained in 3.1. Then, the aforementioned update functions for both phase (ϕ) and frequency (ω), explaining the incorporated level of self awareness as well as how a decentralized collective of pulse copuled oscillators can achieve harmonic synchrony, are thus presented in 3.2 and 3.3.

3.1 System target state: harmonic synchrony

The state of harmonic synchrony is defined [12] as the state in which all agents in the musical collective "fire" or "flash", as described in Subsection 5.1.3, 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 strict synchronization in pulse-coupled oscillators []. How legal

and harmonically synchronized oscillator frequencies and phases look like will be described in this Section.

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. [12] reason while 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 coining a novel state of synchronization for a decentralized system. This novel state arose 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 in a waveform. Such a phenomenon, used as inspiration by Nymoen et al. but not as the basis for any definitions directly, can e.g. be seen in the frequency spectrogram of a humanly hummed G3-tone, depicted in Figure 3.2b. In said example, one can observe the presence of harmonics and overtones having frequencies with integer relationships to the fundamental (smallest) frequency at around 196 Hz, as we see the overtones e.g. has frequencies of around 400Hz, 800Hz, and 1600Hz. We will soon see how such relationships in frequency are used as inpiration for a new definition of a synchronous state; however, the example in Figure 3.2 is only meant to give an idea of where this new state of synchronization stems from, and not as a visual representation of said novel synchrony state.

More accurately then and inspired by integer relationships in waveforms, although not exactly analogus to the example above in Figure 3.2, Nymoen et al. describe *legal* frequencies the oscillator frequencies has to lie within to be considered *harmonically synchronized*:

Legal frequencies definition:

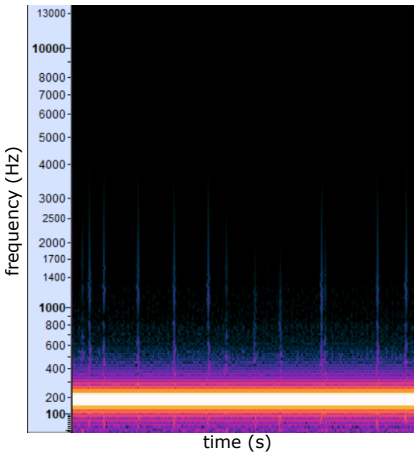
All musical oscillators i, in a harmonically synchronized state, will have frequencies ω_i which are element in the mathematical set

$$\Omega_{legal}(\omega_0) = \omega_0 \cdot 2^{\mathbb{N}_0} = \{\omega_0, 2\omega_0, 4\omega_0, 8\omega_0, ...\}, \tag{3.1}$$

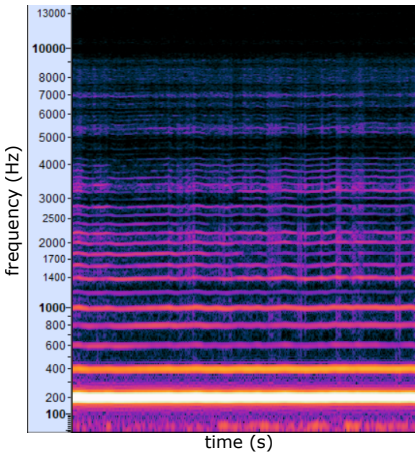
where ω_0 is the lowest frequency in the oscillator collective (or the fundamental frequency if you will), and $\mathbb{N}_0$ are the natural numbers including the

[2] `https://www.szynalski.com/tone-generator/` (accessed 2022.05.17)

[3] `https://github.com/theRealSherapat/CompSA/blob/sa-scopes-implemented/CollectedData/Audio/G3_196Hz_humming.wav` (accessed 2022.05.17)



(a) The frequency spectrogram of the audible waveform being a monotone and purely generated[2]G3-tone at 195.99 Hz.



(b) The frequency spectrogram of the audible waveform being a more-or-less monotone but non-pure G3-tone, hummed and recorded[3]by me, as I tried to repeat the tone in 3.2a with my voice.

Figure 3.2: 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 3.2a and 3.2b respectively, as well as the integer-relationships between the fundamental (lowest) frequency and the harmonics in 3.2b. Frequencies in a harmonically synchronized oscillator collective will for the first ϕ-problem resemble the frequencies in 3.2a, 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 oscillator collective will rather resemble, although not completely correspond to, the frequencies in 3.2b, where higher frequencies with integer relationships to the fundamental and lowest frequency can be present at the same time.

number zero.

If e.g. the smallest oscillator frequency in the musical oscillator collective (ω_0) was equal to 1.5Hz, *legal* frequencies the rest of the oscillators in the collective could have would be $\Omega_{legal}(1.5\text{Hz}) = \{1.5\text{Hz}, 3\text{Hz}, 6\text{Hz}, 12\text{Hz}, ...\}$.

Hence, in terms of frequencies ω_i for all oscillators i in the oscillator network, we have *harmonically synchronized* and *legal* oscillator frequencies ω_i if (and only if)

$$\omega_i \in \Omega_{legal}(\omega_0), \forall i. \tag{3.2}$$

Note that the only difference between this definition of *legal oscillator frequencies* and the spectrogram example depicted in Figure 3.2 lies in that for the legal frequencies as defined in (3.1), higher frequencies can only have doubling integer relationships with the lowest fundamendal frequency (e.g. twice or four

times as high), whereas the frequencies of the harmonics seen in the humanly hummed G3 tone (3.2b) additionally had all other positive integer relationships (e.g. three or five times as high) with the fundamental frequency.

Now we will see what this entails for the phases ϕ_i for all the oscillators in the collective.

Legal phases definition:

FYLL INN DEN NYE MATEMATISKE BESKRIVELSEN DU KOM FRAM TIL HER OM DAGEN

This state of *harmonic synchrony* is then the system goal state K. Nymoen et al. achieve using their phase and frequency update functions, as explained above in Section **??** and **??**; and is also the target or goal state we want to continue achieving and experimenting for in this thesis.

3.1.1 Detecting harmonic synchrony

In order to test and evaluate synchronization performance in their firefly-inspired oscillator-system, K. Nymoen et al. [12] develop a measurement used to detect when the system has reached a state of synchrony. Using the firings of the fireflies, some well-defined conditions have to be met in order for the fireflies to be deemed *harmonically synchronized*:

Harmonic synchronizaton conditions:

Condition 1: Firing may only happen within a short time period t_f.

Condition 2: All nodes must have fired at least once during the evaluation period.

Condition 3: Between each t_f, a period t_q without fire events must be equally long k times in a row.

By utilizing transmitted firings or 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 **Harmonic synchrony detection plot** in Figure 5.7 where the oscillators fulfill the abovementioned requirements right before ending the synchronization process.

These requirements, amongst other illustrations in Nymoen et al.'s paper [12], thus constitutes a blueprint for the design of a performance or synchrony measurement 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 or synchrony measurement is implemented for our musical multi-robot oscillator-network, in the synchronization-simulator, will be given in Section 5.4.

3.2 Phase synchronization

3.2.1 Problem statement

In this first and more simple synchronization problem, which we call the **phase (ϕ) synchronization problem**, we assume homogenous and already-synchronized frequencies $\omega_i = 1Hz$ for all oscillators i. The oscillator collective hence only has to synchronize phases ϕ_i for all oscillators i, which are initially uniformly random phase values $\in [0, 1]$. Hence, phase adjustments are needed in order to synchronize phases in the pulse coupled oscillator collective.

3.2.2 Synchronizing phases via phase adjustment

3.2.2.1 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 [10].

Each oscillator gets a new phase, $\phi(t^+) = P(\phi(t))$, accoring to the **phase update function** (3.3) upon perceiving a "fire"-event from one of the other musical nodes:

$$\phi(t^+) = P(\phi(t)) = (1 + \alpha)\phi(t), \tag{3.3}$$

where α is the pulse coupling constant, denoting the strength between nodes [12], t^+ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the oscillator at time t.

So, if e.g. $\alpha = 0.1$, then a musical oscillator's new and updated phase, immediately after hearing a "fire"-signal from another oscillator, 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 oscillator would be "pushed" to fire sooner than it would otherwise (as nodes fire once they have reached phase-climax $\phi(t) = 1$).

3.2.2.2 K. Nymoen's bi-directional phase adjustment

Apart from altering a similar phase adjustment function to the likes of Mirollo-Strogatz's phase adjustment function so that theirs would cooperate well with the system's frequency adjustment in achieving their target goal of *harmonic synchrony* (further explained towards the end of this chapter), K. Nymoen et al. [12] here introduces an example of a bi directional phase adjustment method. The difference between a bi directional method of phase synchronization and a one directional one is that using bi directional phase adjustment, oscillators are simply adjusting each other's phases in an excitatory way; they only "push" other ocillators's phases further or higher (positive updates) when firing themselves, never "holding" or "dragging" them back (negative updates).

Hence, this newer 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, oscillators update their phases with the slightly more complex **phase update function** (3.4) 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:

$$\phi(t^+) = P(\phi(t)) = \phi(t) - \alpha \cdot sin(2\pi\phi(t)) \cdot |sin(2\pi\phi(t))|, \tag{3.4}$$

where t^+ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the oscillator 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 [12].

The effects then of adjusting phases—upon hearing "fire"-events, according to this newest update-function (3.4)—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 (3.4), 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 $\phi(t) \approx 0.5 = \frac{1}{2}$.

3.3 Phase and frequency synchronization

3.3.1 Problem statement

In this second and more challenging synchronization problem, which we call the **phase and frequency (ϕ&ω) synchronization problem**, we no longer assume homogenous and already-synchronized oscillator frequencies like in the first phase (ϕ) synchronization problem when all frequencies were fixed to 1Hz. Now, the oscillators's frequencies are initialized to uniformly random frequencies ω_i within a certain minimum initial frequency ω_{min}^{init} (e.g. 0.5Hz) and maximum initial frequency ω_{max}^{init} (e.g. 4Hz). Hence, the problem is now more complex; the oscillator collective not only has to synchronize all initially random phases ϕ_i as above, but simultaneously, also all initially random frequencies ω_i for all oscillators i as well. Hence, both phase *and* frequency adjustments are needed in order to synchronize both phases ϕ and frequencies ω in the oscillator collective.

3.3.2 Synchronizing frequencies via frequency adjustment

Only one frequency adjustment / update function was used as a starting point for this thesis and eventually implemented in the Unity synchrony simulator. This one frequency synchronization method, as invented by Nymoen et al. [12], is now presented.

K. Nymoen's frequency adjustment

This approach to frequency adjustment in pulse coupled oscillators, in order to achieve synchronized frequencies, stands in contrast to previous approaches of synchronization where oscillators frequencies are either equal and fixed, oscillators have to send out a pulse every oscillator cycle, or they do not include any

explicit self awareness capabilities. This is not the case for Nymoen's frequency adjustment.

In order to achieve *harmonically synchronized* and "legal" oscillator frequencies (cf. 3.1) in the oscillator collective, not only the phases (ϕ) have to be adjusted and eventually synchronized; frequencies (ω) also have to be synchronized through frequency adjustment.

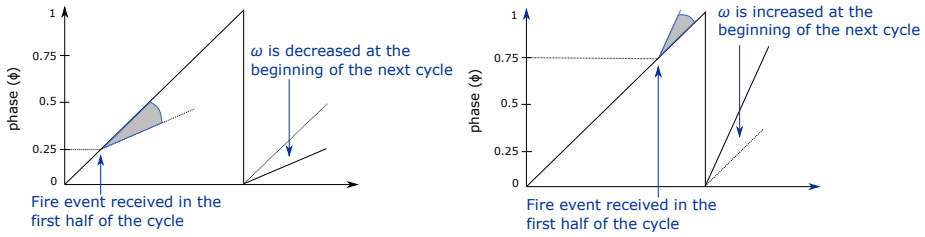


Figure 3.3: An illustration of how frequencies are adjusted after each phase climax (i.e. $\phi = 1$) using Nymoen et al.'s frequency update function. Reproduced from [12].

BESKR.: [OPPRYDDET HITTIL].

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

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{3.5}$$

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

The following steps is how the aforementioned $F(n)$ value is found, and can also be refound in Algorithm 1 and 2 in Chapter 5:

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 value (3.6) is calculated immediately by each oscillator i, having phase $\phi_i(t)$, when a "fire" event signal from another oscillator $j, j \neq i$ is detected by oscillator i at time t.

$$\epsilon(\phi_i(t)) = sin^2(\pi\phi_i(t)) \tag{3.6}$$

As we can see from this error-function, the error-score is close to 0 when the oscillator phase $\phi_i(t)$ is itself close to 0 or 1 (i.e. the oscillator 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 oscillator 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 oscillator with half the frequency of the faster oscillator would then have phase $\phi(t) = 0.5$ when it would hear the faster oscillator "fire"/"flash" — leading to its Error-score $\epsilon(0.5) = sin^2(\pi/2) = 1$, which then makes it seem like the slower oscillator 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, 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* $s(n)$.

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 oscillator is out of synch. Conversely, if we have a low $s(n)$-score, indicating mainly low error-scores for the oscillator — then we have an indication that the oscillator is in synch, hence leading to low error scores, and in turn low $s(n)$-scores.

In other words, each oscillator 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 level of 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.

$$\rho(\phi) = -sin(2\pi\phi(t)) \in [-1, 1] \tag{3.7}$$

For example, if an oscillator i has phase $\phi_i(t) = 1/4$, it gets a value $\rho(1/4) = -sin(\pi/2) = -1$; meaning, the oscillator's frequency should be decreased (with the highest amplitude actually) in order to "slow down" to wait for the other nodes. Conversely, if an oscillator j has phase $\phi_j(t) = 3/4$, it gets a value $\rho(3/4) = -sin(3/2\pi) = -(-1) = 1$; meaning, the oscillator'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, acting as "frequency update contributions", is then as previously mentioned calculated immediately after the oscillator perceives another oscillator's "flashing" signal:

$$H = \rho \cdot s \tag{3.8}$$

Here we then multiply the factor $\rho(n)$ representing how much, as well as in which direction, the oscillator should adjust its frequency, together with a factor $s \in [0, 1]$ of the adjusting oscillator's self-assessed synch-score. This implies that $H \in [0, \rho(n)]$. We hence see that the smallest value $H(n)$ can take for the nth "fire"-event is -1, which it does when $\phi = 0.25$ and $s = 1$. The highest value it can take is 1, which it does when $\phi = 0.75$ and $s = 1$. 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 (3.6)) — 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 oscillator which will hear "fire"-events consistently when it has its phase $\phi \approx 0.5$ (if the agents are synchronized) will, even though it gets a high *out-of-synch score* $s \approx 1$, 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."

To recall, the self-assessed synch-score $s(n)$ tells an adjusting oscillator 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 oscillator 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 oscillator was at the time the oscillator'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.

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

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{3.9}$$

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

$$F(n) = \beta \sum_{x=0}^{y-1} \frac{H(n-x)}{y}, \tag{3.10}$$

where β is the frequency coupling constant, y is the number of heard/received "fire-event"s from the start of the last oscillator 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 (3.10), is a weighted average of all the oscillators's $H(n)$-values accumulated throughout the oscillator's last cycle.

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].

4.1 Software

- Notepad++ v8.1.9.2 (64 bit). For writing my master's thesis and code.
- C#.
- 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 audible waveforms, as in Figure 2.6, as well as frequency spectra of such recorded waveforms, as in Figure 3.2.
- LMMS 1.2.2, being a music-making system and digital synthesizer, for synthesizing harp-tones used for designing "fire"-signals.

4.2 Hardware

- CXT USB-Microphone for recording the hummed waveform, as in Figure 3.2b.

Chapter 5

Implementation and experimental setup

BESKR.: [What I've done / developed].

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 measurement used to evaluate these methods with. 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 3.1.

These goals firstly require of the agents the modelling of oscillators with their properties, like phase and frequency, as explained further in Subsection 5.1.2. 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 5.1.3.

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 og dets system-komponenter du har utviklet i Unity selv, veldig gjerne med et fint Unity-/Simulator-system-skjema].

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" [12]. 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.

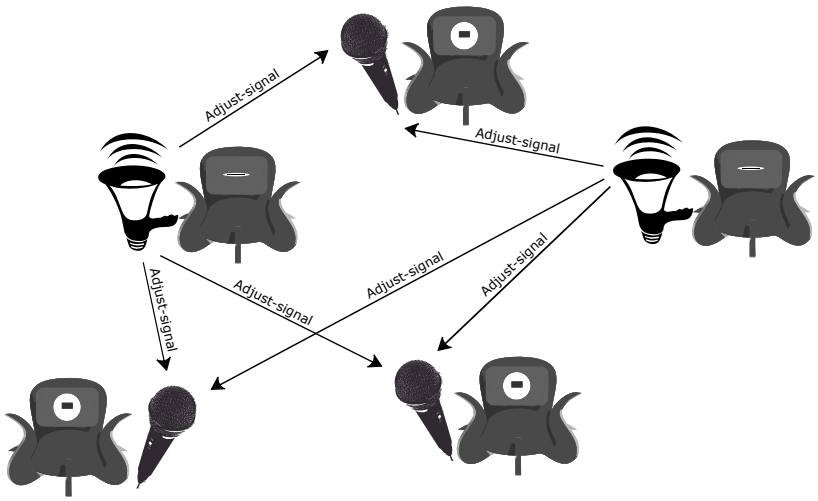
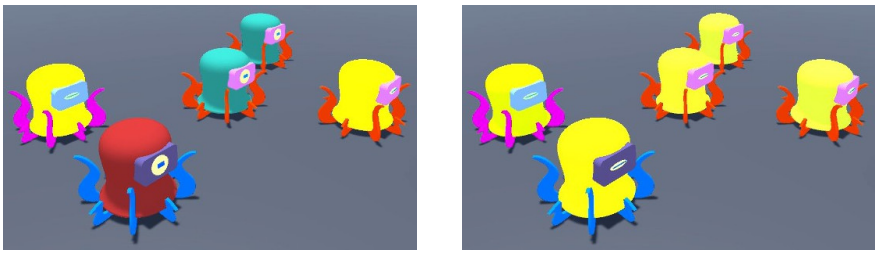


Figure 5.1: Illustration/Schematic: The musical robot collective entraining to synchronize to each other, or more specifically to achieve harmonic synchronization, through performing phase- & frequency-adjustments. Agents that are not firing at the moment will adjust themselves after hearing a transmitted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

Now, the synchrony simulator system components will be expounded and explained in detail.

5.1.1 The simulator environment and its hyperparameters

Hyperparameters further explained



(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.

t_{ref}^{dyn}: Roboticists like e.g. K. Konishi and H. Kokame [6] suggest that by reducing the time oscillators in wireless sensor networks are active (i.e. increasing inactive time), power usage can also be reduced—something which makes oscillator systems last longer as well as being better for the environment in the end. Hence, a as large t_{ref} or t_{ref}^{dyn} as possible, seems to have some real benefits in real systems.

5.1.2 The individual agent: a musical robot

As introduced and presented earlier in 2.7.2, our musical robot collective will then consist of models of M. J. Krzyzaniak and RITMO's *Dr. Squiggles*.

The aforementioned (cf. 2.7.3) 3D-models of these Dr. Squiggles robots for simulation are reused, with permission, for the simulation system designed in this thesis project. They 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 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

Notation	***Definition***
r_i	Individual musical robot i.
R	Musical robot collective consisting of all robots r_i.
ϕ_i	Phase of r_i's oscillator component.
ω_i	Frequency of r_i's oscillator component.
sim s	Simulation seconds. The unit of measurement, w.r.t. to time, used in the thesis and simulation. Simulation time does not always correspond to real / physical time, depending on how fast the hardware is able to run the simulations.

Table 5.1: Notation and terminology used throughout the thesis for system components.

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.

Unless otherwise is stated, the heterogenous visual looks of the Dr. Squiggles robots in Unity have no real difference in the simulator and is only that (for visual looks), not implying other values or methods used.

5.1.3 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. [12]) to attain the system target goal of *harmonic synchrony*, to be elaborated upon in Section 3.1.

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.

5.1.3.1 Under the hood in the simulator

Three distinct *self awareness scope* scenarios are implemented for each individual musical robot in Unity:

Hyperparameter	***Meaning***
$\|R\|$	Musical robot collective size, where R (as defined above) is the set of all musical robots r_i , i = 1, 2, ... , $\|R\|$.
t_{ref}	The time (sim s) robots are unresponsive to adjusting themselves (as a consequence of hearing other robots firing) directly after firing themselves.
t_{ref}^{dyn}	Dynamic refractory period. Same as t_{ref}, only that now the refractory period is given as the percentage of robot r_i's oscillator period (e.g. 10% of $1/4Hz = 0.25s$, i.e. $0.1{*}0.25s = 25ms$, if the oscillator frequency $\omega_i = 4Hz$).
t_f	Legal firing time during which robots r_i are allowed to fire when being detected harmonic synchrony for.
k	even beat counter.
t_{max}	Maximum time limit given to musical robot collectives during which the collective might achieve the target state of harmonic synchrony during, unless they are terminated and deemed "synchronization fails."
ω_{min}^{init}	Minimum initial oscillator frequency a robot's frequency can be assigned at the start of simulations.
ω_{max}^{init}	Maximum initial oscillator frequency a robot's frequency can be assigned at the start of simulations.

Table 5.2: The environmental and collective hyperparameters present and used for the synchronization simulator in Unity.

Hyperparameter	***Meaning***
α	Phase coupling constant. The larger the α, the more a robot adjusts its phase when hearing "adjustment signals" from neighbouring robots.

Table 5.3: The hyperparameters possible to alter for the individual musical robots in the synchronization simulator in Unity.

1. k_s **nearest neighbours self awareness scope**: Each individual robot hears the k_s nearest neighbouring robot's "fire" signals (cf. 5.1.3). In this scenario, at least for small k_s values, robots can have more limited and local knowledge.

2. **Radial** d_s **self awareness scope**: Each individual robot hears neigbouring robots's "fire" signals within a radius d_s around it. In this scenario, robots's self awareness scope is more focussed on the spatial locations of other robots, and can also simulate the robot only hearing robots close in space to itself.

3. **Global self awareness scope**: Each individual robot hears all other neigbouring robots's "fire" signals. In this scenario, robots have maximum and global knowledge when it comes to awareness of other neighbouring robots.

In this way, the degree to which robots are self aware of or communicating

with other robots is increasing. The effects of increasing or decreasing this degree of self awareness are shown in Chapter 6, and especially in 6.2.5 and 6.3.3.

5.1.3.2 Audible to human simulator-observer

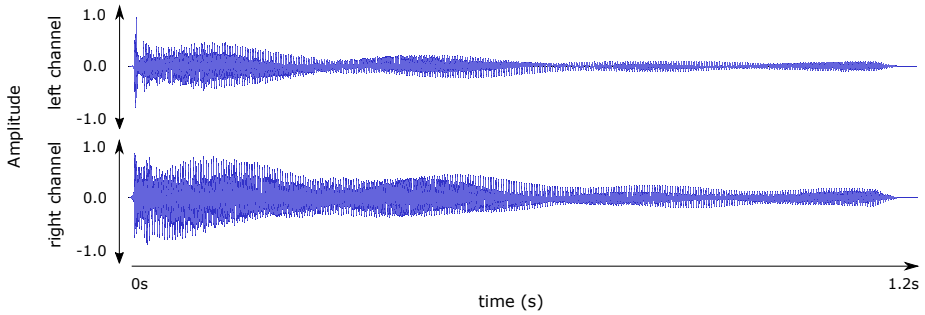
Fire signal design The fire-signal audible to the human observer watching and listening to the musical robots synchronizing to each other should be distinct and short enough so that one can clearly distinguish between which robots are firing and when. At the same time, the firing-sounds should to the human observer not be too sharp and loud to listen to so that it is uncomfortable observing the musical robot collective. Keeping these aspects in mind, some relatively soft and pleasant firing-sounds from an instrument usually associated with harmonious music—the harp—were produced and developed for the synchronization-simulator.

Some manually and empirically perceived harmonious (in terms of sounding good when played together) musical tones were digitally reproduced and recorded in the form of harp-plucks. This was achieved using the music-making system and digital synthesizer LMMS, and a digital string-instrument in the form of a LMMS-plugin from DSK Music[1]. The musical tones, perceived to be harmonious when being heard simultaneously and reproduced with the digital harp instrument, were some of the very first tones perceived in the intro of Pogo's song "Strangerous[2]." One possible avenue to explore in order to find harmonious chords or tones—when played together—in a more automatic approach, live and online during simulation, is discussed in Further work in Chapter 7.

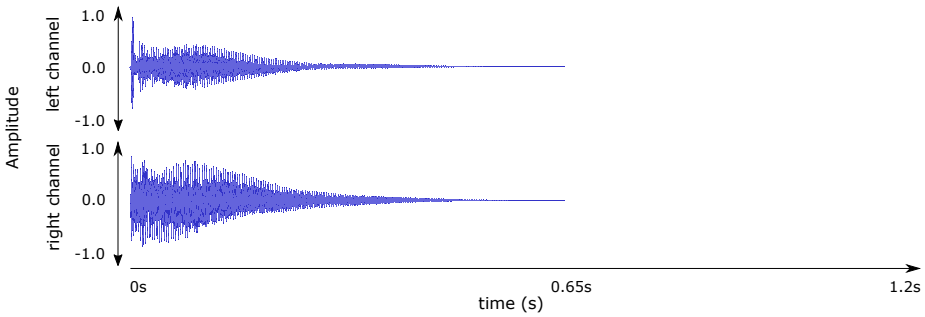
However, the harp-sounds produced with DSK's LMMS-plugin are primarily long and constant harp-plucks, as shown in Figure 5.3a. Using such harp-plucks as a fire-sound directly can make it hard to distinguish to the human ear when multiple robots are playing these frequently and simultaneously. The harp-sounds thus need to be slightly edited—which they in our implementation were in the audio-editing program Audacity—so that the long and constant harp-pluck became more of a "quick" and distinguishable "sound-bullet" — essentially what we want in a solid "fire"-signal sound. The "before" and "after" of such an audio-editing process is depicted in Figure 5.3. Edits performed on waveform 5.3a to obtain waveform 5.3b consists of effects like "Fade Out" (to dampen the sound in the tail of the waveform **INKL.:** [and get a shorter "sustain"]**?**, and hence obtaining a "quicker" sound), in the complete beginning "Amplify" (in order to get a as high "attack"/max-amplitude as wanted to make the fire-sound audible enough before it quickly decays) — and obviously cutting the waveform accordingly (from where it had amplitude ≈ 0 to the end of the waveform).

Frequency based fire-sound assignment **BESKR.:** [Presenting the assignment of firing-sounds, as developed in **Fire signal design**, (with varying fundamental frequencies) based on robot-frequencies].

[1] `https://www.dskmusic.com/dsk-world-stringz-updated/` (accessed 2022.05.17)
[2] `https://www.youtube.com/watch?v=cRzcsXDBn8g` (accessed 2022.05.17)



(a) A harp-sound (stereo audio waveform).



(b) A fire-sound transformed/designed harp-sound.

Figure 5.3: Here we see the starting point and end result (before and after) of a fire-sound design process, where the longer harp-sound in 5.3a got edited and transformed into the more rapid—and amongst many other sounds like it played at the same time, more distinguishable—fire-sound in 5.3b.

GJØR: [Lag kjapt en skisse som i Logs&History-rM-notatet og kombiner dee med implementasjons-spesifikke detaljer som hvordan jeg fikk til å bytte frekvens].

Given that our developed system is not simply a synchronization-system, which could have been implemented with simpler electrical circuits e.g. [], but also a musical robot system — the displayal and expression of the robots synchronizing and becoming synchronized should have a musical dimension, as well as providing a clear way to see whether the robots are getting synchronized or not. Therefore, a musically interesting and meaningful way of signalizing the synchronization-process was needed. The answers to the question of what is different between robots throughout the simulation plays a key part here; namely, e.g. the robot's oscillator-frequencies. Different musical tones were recorded (again with a harp-instrument) and later edited into "fire"-signals, as described above, with various signal lengths given their pitch. The idea was to assign, online and dynamically throughout the simulation-run, "fire"-signal-transformed musical tones with higher pitch (or waveform-frequencies) to musical robots with the highest oscillator-frequencies in the robot-collective.

5.2 Synchronizing oscillator-phases

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 3.2.2.1, is implemented in the Unity simulator, and each agent is en-

dowed with **phase update function** (3.3) 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 5.4.

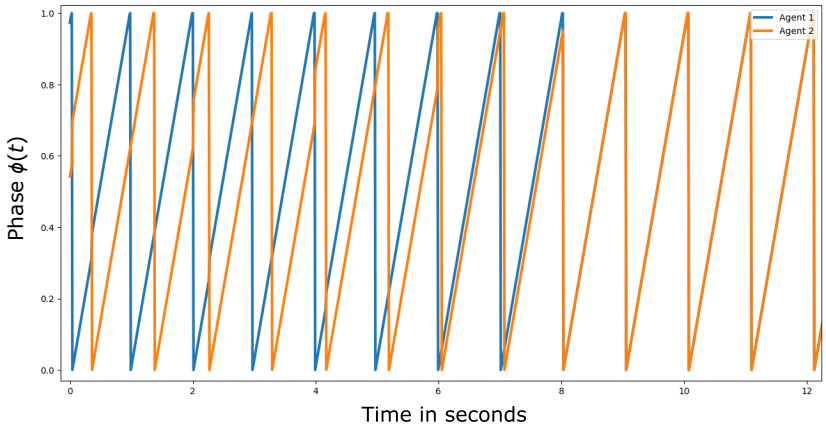


Figure 5.4: "Standard" phase-adjustment with Mirollo-Strogatz's approach

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 **??**, is implemented in Unity, and each agent is endowed with **phase update function** (3.4) 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 5.5.

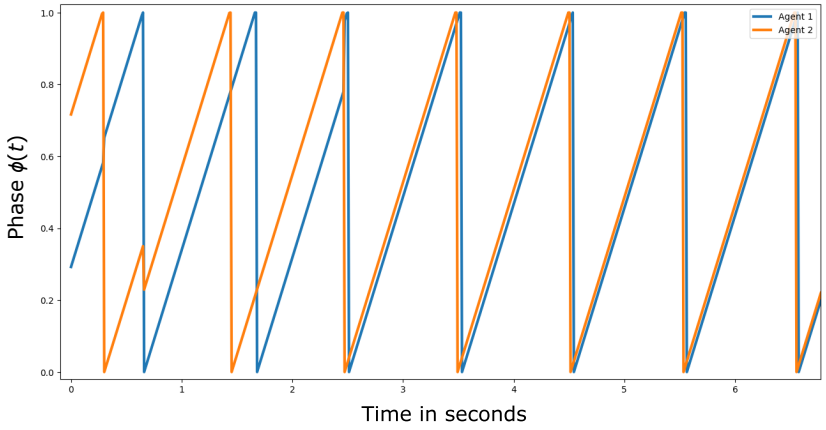


Figure 5.5: Bi-directional phase-adjustment with K. Nymoen et al.'s approach

5.3 Synchronizing oscillator-frequencies

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 3.1, 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 3.1.

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

The details of how the Dr. Squiggles in Unity adjust and eventually synchronize (harmonically) their oscillator frequencies, is shown in Algorithm 1 and 2. Algorithm 1 outlines how the "frequency adjustment contributions" H get calculated as soon as the Dr. Squiggles hear a neighbour Dr. Squiggle sending a "fire" signal; whereas Algorithm 2 shows how these previously calculated $H(n)$ values are used to obtain the new and updated oscillator frequency $\omega_i(t^+)$ with which individual oscillators assign their frequencies to after each phase climax (i.e. when $\phi = 1$).

Algorithm 1: Calculating frequency update contributions H in Unity

procedure $(\phi(t))$
  if not in refractory period **then**
    $\epsilon \leftarrow \sin(\pi\phi(t))^2$ ▷ Step 1
  else
    $\epsilon \leftarrow 0$
  end if
  ϵ is added to $errorBuffer$
  $s \leftarrow \text{median}\{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}$ ▷ Step 2
  $\rho \leftarrow -\sin(2\pi\phi(t))$ ▷ Step 3
  $H \leftarrow \rho \cdot s$ ▷ Step 4
  H is added to list $H(n)$
end procedure

5.3.1 Verifying K. Nymoen's frequency-adjustment

In the newly proposed Unity simulator environment, the previously introduced self-assessed sync score $s(n)$ (in 3.3.2) is implemented as the median of a list

Algorithm 2: K. Nymoen et al.'s frequency updates in Dr. Squiggles robots

procedure $(H(n), \beta)$
  $cycleHSum \leftarrow 0$ ▷ Step 5 started
  $y \leftarrow \text{length}(H(n))$
  for all $H \in H(n)$ **do**
    $cycleHSum \leftarrow cycleHSum + H$
  end for
  $F(n) \leftarrow 0$
  if y not 0 **then**
    $F(n) \leftarrow \beta \cdot cycleHSum/y$ ▷ Step 5 finished
  end if
  Emptying $H(n)$ ▷ for new contributions next oscillator cycle
  $\omega_i(t^+) \leftarrow \omega_i(t) \cdot 2^{F(n)}$
end procedure

containing m error-scores ϵ. Such an error score list of length m is easily implemented by a list called *errorBuffer*:

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

then leading to:

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

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 3.3.2) 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 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 period.

By using K. Nymoen et al.'s frequency adjustment method, synchronization of initially random frequencies in our newly developed Unity simulator is achieved, like can be seen recorded in Figure 5.6.

5.4 Detecting harmonic synchrony

The performance measurement will be used in our synchrony-simulator to evaluate and test the multi-robot collective's ability to harmonically synchronize to each other. As mentioned in Subsection **??**, K. Nymoen et al.'s requirements and illustrations [12] for achieving *harmonic synchrony* serve as a blueprint or

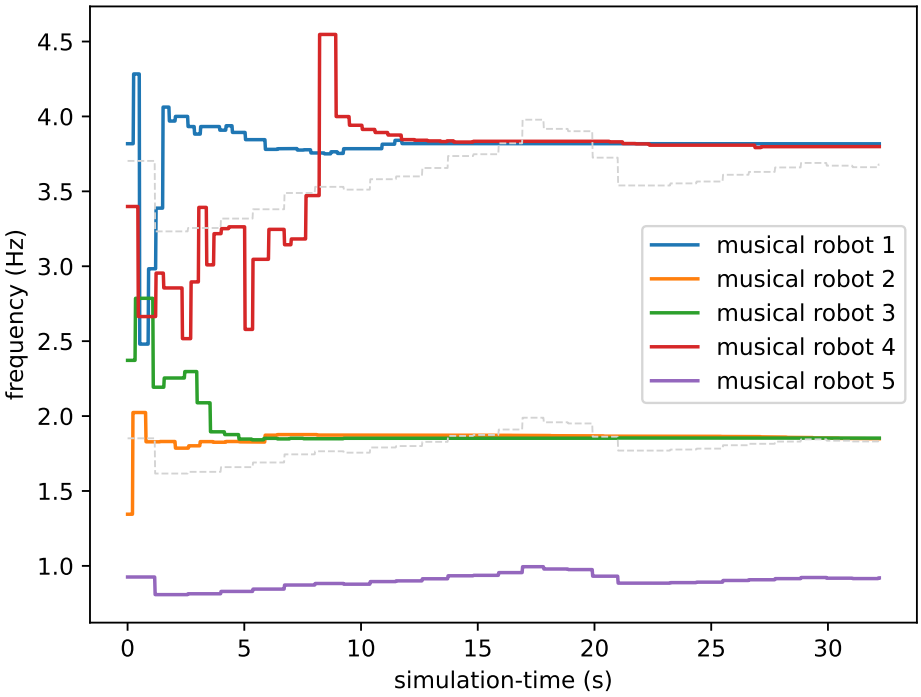


Figure 5.6: Frequency synchronization for five robots in a Unity synchronization simulation run achieving harmonic synchrony after 30 seconds. Note how the legal harmonic frequencies (dashed gray lines) are defined by the lowest—fundamental—frequency ω_0 in the robot collective fluctuating slightly below 1Hz, correctly leading to the legal frequencies right below 2 and 4 Hz.

guide for how to similarly implement our synchrony or performance measurement. This performance measurement should be able, during synchronization-simulation, to 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 5.7. In this figure, one can see at which times (sim s) the musical robots fired during a synchrony simulation run ending up in either harmonic synchrony, or as a "synchronization fail."

The resulting and corresponding performance scores obtained using this performance measurement will then take values of the simulation time (s) 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 3.1.

Now if **Conditions 1-3** from Subsection **??** are kept, we have harmonic synchrony.

My specific implementation of the synchrony measurement essentially consists of enforcing all the requirements or rules listed in **??**, given some constant t_f- and k-values (e.g. $80ms$ and 8 respectively [12]). And again—to recall from **??**—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 occuring several times in Figure 5.8. Note that in this specific simulation run above, the agents were on their way to achieve harmonic synchrony five times before the 10th second of the synchronization-simulation 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 in this case 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. Note that this gives us a sense of how synchronized the robot collective is over time; the more even beats the robots have in a row, the more synchronized they are.

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, given by the following formula, which is visually explained further in the schema in Figure **??**:

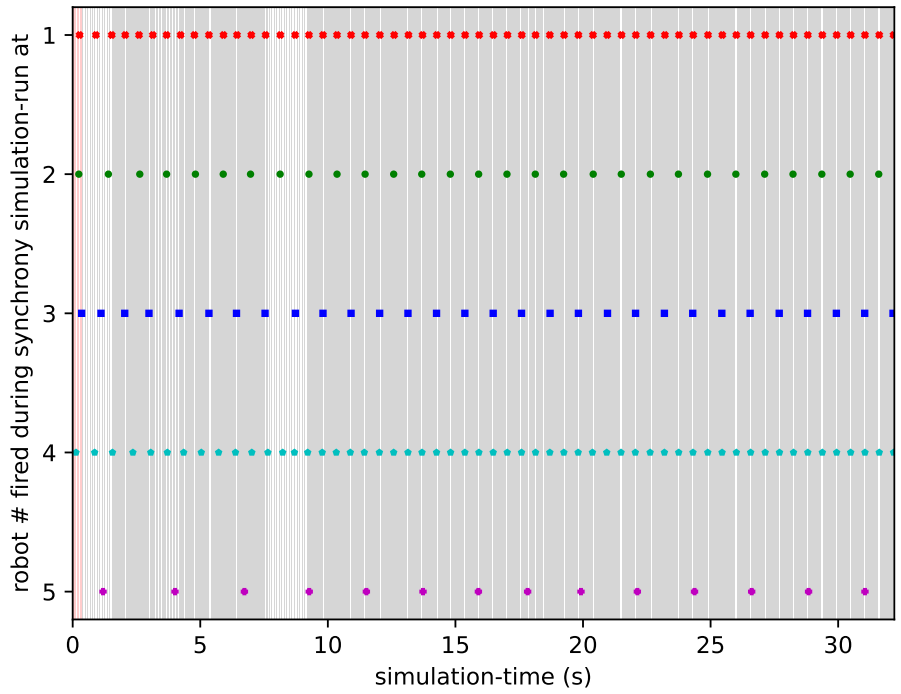


Figure 5.7: **Harmonic synchrony detection plot**: recording a detection of harmonic synchrony (hvordan harmonic synch. ble detectet i et simulation-run). The steps and events of **I)** through **VII)** can also be seen commented in pseudocode []. Initial red area represents the "start-up"-period where no (gray) t_q-value is defined yet. **I)** First robot firing, initiating a process of defining such a "silent" time-window t_q within which no nodes are allowed to fire, and a whole "legal-firing" t_f-window (here shaded in white and only 80ms long, as in [12]) ensues. **II)** At least one robot firing, and with their firing-time(s) giving us the *early* t_q*-defining* time-value (see schema below), as well as triggering second whole t_f-window to ensue. **III)** At least one robot firing after the *early* t_q*-defining* time-value was found, hence (with their firing-time(s)) giving us the *late* t_q*-defining* time-value right before using these when finishing the on-started process of defining the "silent" time-window t_q; triggering a half (due to stabilization purposes by future centering of fire-events) t_f-window to ensue. **IV)** A robot is caught firing illegally during a "silent" t_q-window and hence resets the *towards_k_counter*-variable, as well as (re-) starts a process like in **I)** for (re-) defining a new t_q-value. **V)** Firing after this point only happens during short "legal-firing" t_f-windows, so **Condition 1** (cf. **??**) is held. **VI)** All robots have fired at least once during the evaluation-period, so **Condition 2** is held. **VII)** The robot collective has fired legally and evenly, without resetting t_q, k (equal to 8 in this case) times in a row, so **Condition 3** is fulfilled — and harmonic synchrony is thus achieved and detected after 21.9 seconds of synchronization.

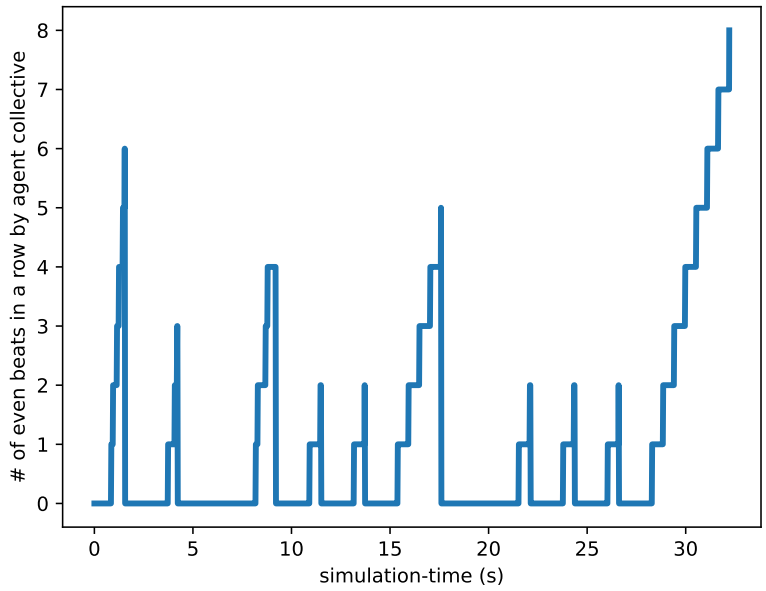


Figure 5.8: A **synchrony evolution plot**, displaying the temporal recording of the *towards_k_counter* variable throughout a synchrony simulation-run in Unity. The counter is incremented as the robot collective fires evenly within 'legal' t_f windows, and is conversely reset to 0 if illegal firings during 'silent' t_q windows are heard.

$$\begin{aligned} t_q^* &= late\_t_q^*\_defining\_timevalue - early\_t_q^*\_defining\_timevalue - t_f \\ &= median(t_1^{'}, t_2^{'}, ..., t_m^{'}) - median(t_1, t_2, ..., t_n) - t_f, \end{aligned} \tag{5.3}$$

where n fire-events were recorded during the earlier t_f-window, and m fire-events were recorded during the later t_f-window.

The specific steps and procedures needed to implement this harmonic synchrony detection is expounded in detail in Algorithms 3, 4, and 5.

Algorithm 3: Harmonic synchrony detection part A (må evt. fylles inn)

Algorithm 4: Harmonic synchrony detection part B (må evt. fylles inn)

If a certain amount of time, e.g. 5 simulation minutes [12], has gone without the detection of harmonic synchrony occuring, the simulation-run is terminated as a "fail".

Algorithm 5: Harmonic synchrony detection part C (må evt. fylles inn)

Chapter 6

Experiments and results

BESKR.: [Gode eksperimenter som motiveres og forklares, settes opp, og utføres m/resultater man diskuterer og analyserer. Det er bra å evaluere fra flere synspunkt med flere research methods og hvis tid].

This chapter presents the experiments set up and performed in the novel synchronization simulator in Unity, as presented in Chapter 5, for certain configurations of musical robot collectives. Effects of the individual musical robots's hyperparameters on the collective achievement and performance of achieving harmonic synchrony are presented. Some examples are hyperparameters which determines how much each musical robot will adjust itself after hearing a transmitted fire signal from a neighbouring robot; α for phase adjustment and β for frequency adjustment.

The main performance scores presented in this chapter will consist of synchronization times given in simulation time (s) (i.e. how long it takes robot collectives to reach the state of harmonic synchrony if they ever do), accompanied by the respective and corresponding error rates during the belonging simulation runs (i.e. the percentage of robot collectives out of e.g. 30 runs failing to reach harmonic synchrony before the maximum time limit of e.g. 5 simulation minutes). If a musical robot collective then never reaches the target state of harmonic synchrony within the maximum time limit, this simulation run will be regarded as a "synchronization fail", and we will then not regard its termination time (in simulation time seconds) as *harmonic synchronization time*—but simply that, that simulation run's termination time.

As previously mentioned, relating to the different robot colors like turquoise, red, and green, all robots are homogenous apartly from the visual; all individual robots have for experiments the same hyperparameters unless otherwise is explicitly stated.

6.1 Data visualization

BESKR.: [Der jeg forklarer SimRun-plotsa mine som analyserer synkroniseringsruns, som jeg kun skal bruke samlet med screenshots av synkroniseringsruns før vs. etter én gang for 1) Phase sync og 2) Phase and frequency sync, og ellers

bare evt. (sub-)plots av dersom det er hensiktsmessig og ikke overflødig og rotete].

6.2 Phase synchronization

This is the section where experiments attempting to synchronize for the first and simpler problem, namely synchronizing only the phases ϕ_i of all agents i, are presented and analyzed resuls for. These are then experiments where all musical robots have an equal and fixed frequency, only adjusting phases, in order to entrain to synchronize their phases to each other until reaching the target state of harmonic synchrony. Or in other words, all experiments in this phase synchronization section have the two hyperparameters shown in Table 6.1.

Adj_ϕ	Adj_ω
Mirollo & Strogatz	None

Table 6.1: Phase synchronization experiments hyperparameter setup.

6.2.1 Initial validation

To see that synchronization is actually happening like intended, some relevant quantities and simulation values are recorded for a phase synchronization simulation run in Unity and displayed in **??**.

FYLL INN.

6.2.2 Reproducing baseline results

Experiment setup: In order to see that our developed synchronization simulator in Unity yields more or less the same results as Nymoen's results, similar experiments as reported in their paper are performed here. These tell us whether differences in performance, in terms of synchronization times (sim s), is simply due to implementation differences, or actually because of the utilized synchronization methods and hyperparameters in question.

First off, Mirollo & Strogatz's phase adjustment method (as presented in 3.2.2.1) is experimented with for the initial phase (ϕ) synchronization problem, for varying phase coupling constants α. Hyperparameters are set in the simulator as shown in Table 6.2. Results can be seen in Figure 6.1.

$\|R\|$	β	t_{ref}	t_f	k	t_{max}
6	0	50ms	80ms	8	5min

Table 6.2: Experiment hyperparameter setup.

Experiment results:

Experiment analysis:

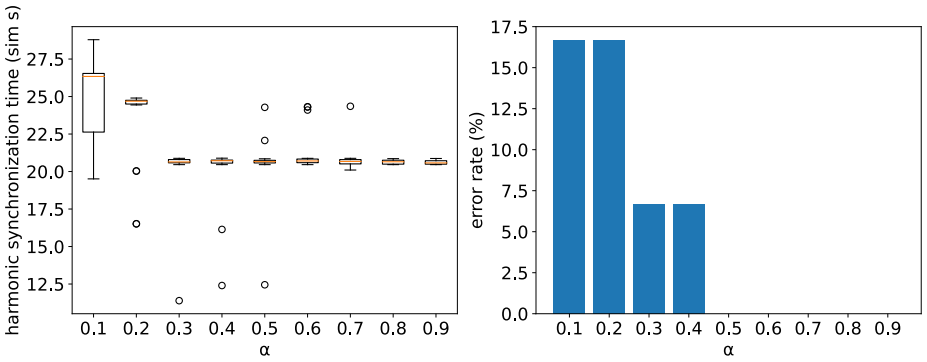


Figure 6.1: Harmonic synchronization times (s) for 6 robots with initially random and unsynchronized phases but equal and fixed frequencies (1Hz), for varying phase coupling constants α. 30 simulation runs per α are reported.

6.2.3 Hyperparameter tuning

Experiment setup: Here we tune the hyperparameters α and t_{ref}^{dyn} for several robot collective sizes, according to performance scores of how long it takes robot collectives on average to achieve harmonic synchrony. Exactly these two specific hyperparameters are experimented with mostly since they empirically seem to be the most important ones to set correctly before starting the simulator; that is, in order for the robots to actually manage achieving harmonic synchrony. Additionaly, the point of K. Konishi and H. Kokame (cf. 5.1.1) was also remembered.

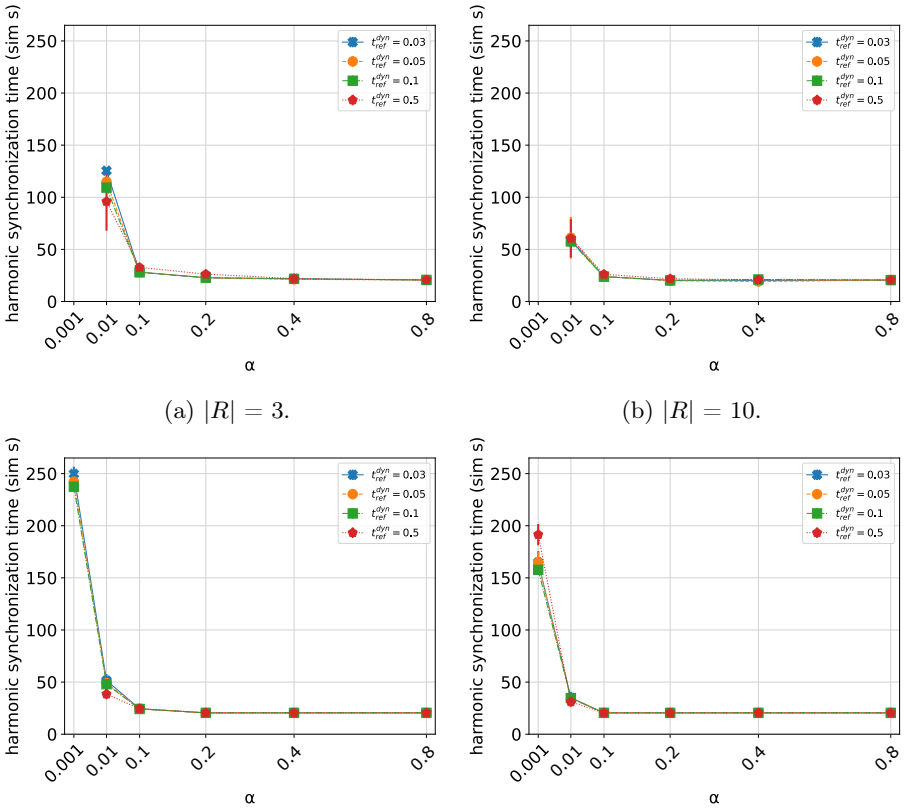
The specific values of hyperparameters to test synchronization times for were chosen based on an initial and identical but failed experiment (not reported) where the tested values for t_{ref}^{dyn} were showing some interesting effects for various robot collective sizes. Simple and limited insight was also collected from trial and error in the complete beginning when trying to facilitate stable and successful simulation runs. And furthermore, after seeing results found in this thesis for α already, we also got some more ideas for which values of α could be interesting to investigate further.

Other constant hyperparameters not being tuned or experimented for in this experiment are shown in Table 6.3. The hyperparameter tuning results, where the effects of tuning hyperparameters α and t_{ref}^{dyn} for various musical robot collective sizes, are shown in Figure 6.2. Again, since we are synchronizing for the phase (ϕ) synchronization problem, now only phases are initially unsynchronized, and frequencies are fixed and constant (1Hz) throughout the simulation runs.

β	t_f	k	t_{max}
0	80ms	8	5min

Table 6.3: Experiment hyperparameter setup.

Experiment results: As we can see in Figure 6.2, there are generally low standard deviations which make errors in the plots barely visible. We also



(a) $|R| = 3$.

(b) $|R| = 10$.

(c) $|R| = 25$.

(d) $|R| = 50$.

Figure 6.2: Average harmonic synchronization times are plotted in errorbar plots, where the standard deviation is the error. 30 simulation runs per α and t_{ref}^{dyn} pair are reported, unless simulation runs ended up as "synchronization fails."

can notice that the largest differences are seen for lower α values, as average harmonic synchronization times are more and more overlapping and similar for larger α values.

A general pattern we can see is that larger robot collective sizes (larger $|R|$ values) handle lower phase coupling constants α better; in that the music collectives both achieve harmonic synchrony in lower average times than the smaller robot collectives, as well as achieving lower error rates. To this latter point, note that for the lowest α value of 0.001, robot collectives with size $|R| = 3$ and 10 were not able to achieve harmonic synchrony at all during any of the simulation runs (i.e. having 100% error rate), whereas larger collective sizes like $|R| = 25$ and $|R| = 50$ managed to achieve harmonic synchrony despite the low α value.

Experiment analysis: The explanation for this latter observation regarding the seemingly increasing robustness with increasing collective sizes $|R|$ can very well lie in the fact that, as we remember, the phase coupling constant α repre-

sents how much each robot will adjust its phase when hearing a "fire" signal from a neighbouring robot. If there are more neighbours firing "adjustment signals" to a certain robot (i.e. we have a higher collective size $|R|$), it is also logical that the robot in question will update its phase more often. And so we can then see why e.g. all average harmonic synchronization times for $\alpha = 0.01$ seem to improve (i.e. gets reduced) for every increase in collective size $|R|$; the phase coupling constant $\alpha = 0.01$ might be weak, but with further and more frequent weak adjustments accumulated over the same time, the weak α is compensated for.

Hence, it seems like larger robot collectives do not require as large of a phase coupling constant α in order to synchronize to each other compared to that which smaller robot collectives require.

6.2.4 Comparing phase adjustment methods

Experiment setup: So far in this chapter and section, we have only run experiments for pure phase synchronization, the ϕ problem, with Mirollo-Strogatz's method of phase adjustment, like described in 3.2.2.1. With this method of phase synchronization, robots are simply adjusting phases in an excitatory way; they only "push" other ocillators's phases further or higher when firing themselves, never "holding" or "dragging" them back.

Now, in order to investigate the validity of the claimed [] benefits of performing bi directional phase adjustments—both excitatory and inhibitory—like that of Nymoen's phase adjustment (see 3.2.2.2), an experiment comparing these two aforementioned phase adjustment methods thus follows.

Empirically speaking based on previous results in phase synchronization experiments, in particular the ones shown in Figure 6.1 and 6.2, arguably the best values so far for α (i.e. $\alpha = 0.8$) and t_{ref}^{dyn} (i.e. $t_{ref}^{dyn} = 0.1$) will be reused here for Mirollo-Strogatz's phase synchronization runs.

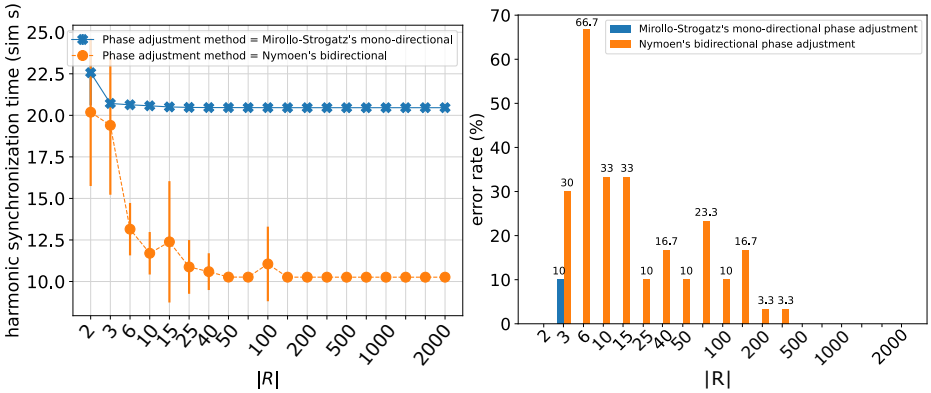
Since we have not yet performed any experiments for pure phase synchronization in the ϕ problem using Nymoen's slightly modified and bi directional phase adjustment function, the same α value of 0.8 will also be used for the phase synchronization runs here.

See the hyperparameter setup below in Table 6.4, and the results for the two phase adjustment methods given various robot collective sizes $|R|$ in 6.3. We here measure how long it takes various sizes of robot collectives to synchronize their phases to each other, given the two different phase adjustment methods.

α	β	t_{ref}^{dyn}	t_f	k	t_{max}
0.8	0	10%	80ms	8	5min

Table 6.4: Experiment hyperparameter setup.

Experiment results: We quickly see from the results in Figure 6.3 there seems to be a considerable difference in harmonic synchronization times between using Mirollo-Strogatz's and Nymoen's phase adjustment methods to synchronize phase in the phase (ϕ) synchronization problem. In Subfigure 6.3a, we see that for smaller musical robot collectives $|R|$, differences in synchronization time on average are not especially large. However, as collective size $|R|$ increases, we



(a) Performance scores.

(b) Error rates.

Figure 6.3: Performance scores 6.3a given in average harmonic synchronization times (sim s) with standard deviation. Corresponding error rates in 6.3b based on 30 individual runs per collective size $|R|$ for each of the two phase adjustment methods.

see the harmonic synchronization times for Nymoen's bi directional phase synchronization becomes considerably lower than that of Mirollo-Strogatz's mono-directional on average, variance—or at least the square root of it—accounted for.

On the other hand, when we look at the error rates in Subfigure 6.3b, we also see that Nymoen's phase adjustment method's error rates are considerably higher than those of Mirollo-Strogatz's error rates (being nearly negligible). Interestingly, this is only the case until collective size $|R| = 500$ though, after the point of which both phase adjustment methods's error rates are both equal to 0, and hence this objection towards Nymoen's phase adjustment method no longer holds. Hence, there seems to—at least up until $|R| = 500$—exist a sort of tradeoff or high risk high reward scenario here, where one can in the phase (ϕ) synchronization problem achieve harmonic synchrony faster by using Nymoen's phase adjustment method while at the same time risking a higher chance of the synchronziation failing; whereas Mirollo-Strogatz's method of phase adjustment on average seems to be the safer, albeit slower, option.

Yet again, we also in these results notice a trend where smaller hyperparameter values lead to more unstable or worse performing synchronizations. Generally, although not strictly nor without exceptions, for both the harmonic synchronization times in Subfigure 6.3a and the error rates in 6.3b, the larger the hyperparameter $|R|$, the more stable or quick synchronization runs we performed in the Unity synchronization simulator.

Experiment analysis: In this specific case, what this latter observation of seemingly increasing robustness given increasing collective sizes $|R|$ points to is perhaps that the system's robustness and stability is increasing as collective size increases—being a strength classically advocated for in the multi agent systems and swarm systems literature []; as well as being found in e.g. robustness degrees in networks where you in the one case have a single point of failure versus many

redundant paths and edges between your nodes [].

As we see, the synchronization simulator in Unity is able to synchronize with a lot of agents, still without having broken the simulation yet. Comparatively, Nymoen et al. [12] only experimented with 6 oscillator nodes in their firefly system.

6.2.5 Increasing degree of self awareness

So far in the harmonic synchronization experiments, we have only and by default used the most restrictive and comprehensive *self awareness scopes*; all individual Dr. Squiggle robots have been globally connected, that is, and always hearing every neighbour Dr. Squiggle's fire signals. Now we want to challenge this strong connectivity assumption.

Information and messages have the ability to travel through networks of many sorts, and the type of connectedness in these networks might influence how this occurs—and indeed the quality or accuracy of the travelled message; e.g. in a game of Chinese whispers versus a message sent through internet directly from the sender to the receiver. Both the speed as well as the overall quality of the message transmission can greatly be decided as a result of how connected the communicating nodes in a network are. Phenomena as these are to be experimented with here, as we will see how the connectedness between the musical Dr. Squiggle robots influence their collective and overall performance in synchronizing (harmonically).

It has previously, although more mathematically and not so much experimentally, been attempted to reduce the connectivity assumptions in pulse coupled oscillators [13] and to analyze the effects of it.

Here the hypothesis of whether increasing musical robots's degrees of self awareness will affect the synchronization performance or not, is investigated experimentally. Exactly what is meant by an increasing degree of self awareness specifically refers to the robots's *self awareness scope* [8]. This is tested in Unity for the more challenging $\phi\&\omega$ problem of harmonically synchronizing both phases and frequencies. Perhaps a larger self awareness scope, meaning more knowledge about the social environment, will lead to the robots having a better "overview" of the environment; hence leading to shorter simulation time (s) before reaching the goal state of harmonic synchrony. Or perhaps hearing more "fire" signals on average simply will be disturbing to the robots and hence disturb and slow down their entrainment towards harmonic synchrony. This experiments attempts to answer questions like these by for the three *self awareness scope* scenarios as introduced in 5.1.3.1 evaluating collective synchronization performance.

6.2.5.1 Self awareness scope tuning

In order to find out whether global connections in the pulse coupled oscillators is necessary to maintain performance in harmonic synchronization, varying *self awareness scopes* (as introduced in 5.1.3.1) are now experimented with.

k_s nearest neighbour scope

Experiment setup: Firstly, the first *self awareness scope* scenario (as in 5.1.3.1) is experimented with and tuned for. The number of neighbouring Dr. Squiggle robots each individual Dr. Squiggle robot will listen for fire signals to (k_s) will be scaled and increased from $k_s = 1$ up until $k_s = |R| - 1$ which is the maximum number of neighbours each Dr. Squiggle robot can listen to for fire signals.

In other words, the individual Dr. Squiggle robots's self awareness scope is increased incrementally in terms of k_s *nearest neighbour SA scope* and evaluated for performance in harmonic synchronization.

Hyperparameters are set in the simulator as shown in Table 6.5. Results can be seen in Figure **??**.

α	β	t_{ref}^{dyn}	t_f	k	t_{max}
0.8	0	10%	80ms	8	5min

Table 6.5: Experiment hyperparameter setup.

FYLL INN

Experiment results:

Experiment analysis:

d_s **radial scope**

6.2.5.2 Simulation breaking

Experiment setup: In order to test the limits of the developed synchronization simulator, we will here investigate the effects of scaling and exploding the musical robot collective size $|R|$—also given some various self awareness scopes found worthwhile to further investigate in the previous experiments in 6.2.5.

6.3 Phase and frequency synchronization

This is the section for the experiments attempting to synchronize for the second and harder problem of synchronizing both phases ϕ_i, as well as frequencies ω_i, for all agents i. These are then experiments where all musical robots originally have unequal and ever-changing phases and frequencies, adjusting both phases and frequencies in order to entrain to synchronize their phases and frequencies until reaching harmonic synchrony. Or in other words, all experiments in this phase and frequency synchronization section have the two hyperparameters shown in Table 6.6.

Adj_ϕ	Adj_ω
Nymoen	Nymoen

Table 6.6: Phase and frequency synchronization experiments hyperparameter setup.

6.3.1 Initial validation

To see that synchronization is actually happening like intended, some relevant quantities and simulation values are recorded for a phase synchronization simulation run in Unity and displayed in **??**.

FYLL INN.

6.3.2 Reproducing baseline results

Again, we want to here see whether we get more or less the same results in the Unity simulator as K. Nymoen et al. get in their firefly oscillator system.

Hence, we here present attempts made in the novel Unity synchronization simulator at recreating K. Nymoen et al.'s first results with their novel frequency synchronization method, which is utilizing, amongst other aspects, self awareness [12].

6.3.2.1 Ordering by phase couplings

Experiment setup: Given that K. Nymoen et al. do not mention their β value in their frequency synchronization experiment where they order for different phase coupling values α, an *empirically decent* β value of 0.4 is chosen for this experiment. What *empirically decent* refers to in this case are K. Nymoen et al.'s findings in the results of their last experiment [12] where synchronization times for various β values were evaluated; deeming $\beta = 0.4$ to be a good value, with no further improvement in synchronization performance when β is increased further.

Here, K. Nymoen et al.'s self aware frequency adjustment method, implemented in our novel synchrony simulator in Unity, is experimented with for varying phase coupling constants α. This time not only oscillator phases are initially unsynchronized; oscillator frequencies are also unsynchronized to begin with. Hence, we here try to synchronize our musical robots in the phase (ϕ) *and* frequency (ω) synchronization problem, using Nymoen's phase adjustment method to synchronize phases, as well as Nymoen's frequency adjustment method to synchronize frequencies. See set up hyperparameters in Table 6.7. See the results in Figure 6.4.

$\|R\|$	β	ω_{min}^{init}	ω_{max}^{init}	m	t_{ref}	t_f	k	t_{max}
6	0.4	0.5Hz	4Hz	5	50ms	80ms	8	5min

Table 6.7: Experiment hyperparameter setup.

Experiment results:

Experiment analysis:

6.3.2.2 Ordering by phase couplings but with more stable hyperparameters

Experiment setup: After seeing how poorly the musical robot collectives managed to achieve harmonic synchrony during previous experiment 6.3.2.1,

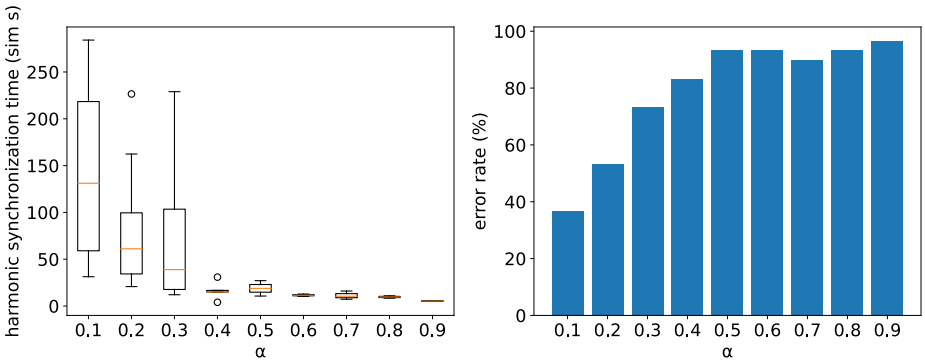


Figure 6.4: Harmonic synchronization times (sim s) for 6 robots with both initially unsynchronized phases *and* frequencies, for varying phase coupling constants α, reaching harmonic synchrony—but also often failing to. 30 simulation runs per α are reported.

we change the hyperparameters slightly in the hopes of achieving harmonic synchrony more frequently, or in other words more stable synchronization simulation runs. The reason is mostly that it would be beneficial to actually see whether we observe similar patterns as in Nymoen's results—something which becomes impossible when robot collectives nearly never achieve harmonic synchrony.

The frequency coupling constant of $\beta = 0.6$ could potentially be another supposed good β value, at least solely judging by Nomoen's results, and hence we select this value for this hopefully stable phase and frequency synchronization experiment.

By initial trial and error, it also became apparent that phase *and* frequency synchronization was not always stable—i.e. the robot collective not managing to achieve harmonic synchrony within the maximum time limit—for collective sizes of 6 or more. It did however become apparent that phase *and* frequency synchronization *was* stable for smaller musical robot collective sizes, like 2 or 3.

Hence, a similar experiment to as in 6.3.2.1 is set up in the Unity synchrony simulator, and is run similarly as before. So still, the phase (ϕ) *and* frequency (ω) synchronization problem is to be experimented for, given different phase coupling constants α, with Nymoen's both phase and frequency adjustment methods—only this time with $\beta = 0.6$ and $collsize = 3$ instead. See set up hyperparameters in Table 6.8. See results in Figure 6.5.

$\|R\|$	β	ω_{min}^{init}	ω_{max}^{init}	m	t_{ref}	t_f	k	t_{max}
3	0.6	0.5Hz	4Hz	5	50ms	80ms	8	5min

Table 6.8: Experiment hyperparameter setup.

Experiment results:

Experiment analysis:

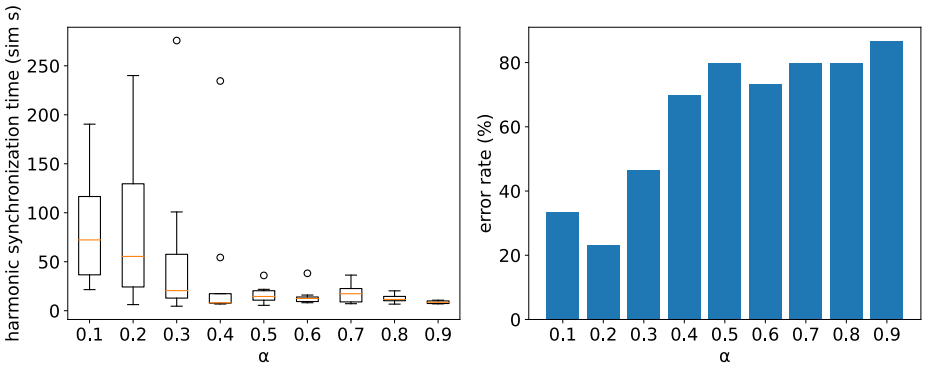


Figure 6.5: Harmonic synchronization times (sim s) for 3 robots with both initially unsynchronized phases *and* frequencies for varying phase coupling constants α. 30 simulation runs per α are reported.

6.3.2.3 Ordering by frequency couplings

Experiment setup: Lastly, for the final Baseline reproducing experiment, it was wanted to see whether results within the phase and frequency ($\phi\&\omega$) synchronization problem, when one varied the frequency coupling constant β, also was more or less in the same ballpark or not, for the same reasons as mentioned in the first Baseline reproducing experiment in Section 6.2.

Since we now do not synchronize in Unity for varying phase coupling constants α, we now fix α and rather test how the individual musical robots's various frequency coupling constants β affect the performance of the musical robot collective.

Again, Nymoen does not specify exactly the phase coupling constant α they use in their latter experiment when testing their firefly-inspired synchronization system for various β values. Hence, the now fixed phase coupling constant α is here selected by reusing a reasonably good α value, based on the similar $\phi\&\omega$ synchronization experiments presented in Figure 6.4 and 6.5. In these two experiments in particular, $\alpha = 2$ yielded both harmonic synchronization times among the best synchronization times, given the error scores; and furthermore, one of the lowest error rates. This then gives us a fixed $\alpha = 0.2$.

See the set up hyperparameters for the experiment in Table 6.9, and the corresponding results in Figure 6.6.

$\|R\|$	α	ω_{min}^{init}	ω_{max}^{init}	m	t_{ref}	t_f	k	t_{max}
6	0.2	0.5Hz	4Hz	5	50ms	80ms	8	5min

Table 6.9: Experiment hyperparameter setup.

Experiment results: Even though we do not see exactly the same harmonic synchronization times as we see in Nymoen's results, and worse at that, we do in fact see a similar pattern in that also here, synchronization times and error scores seem to improve the larger frequency coupling constant β we have.

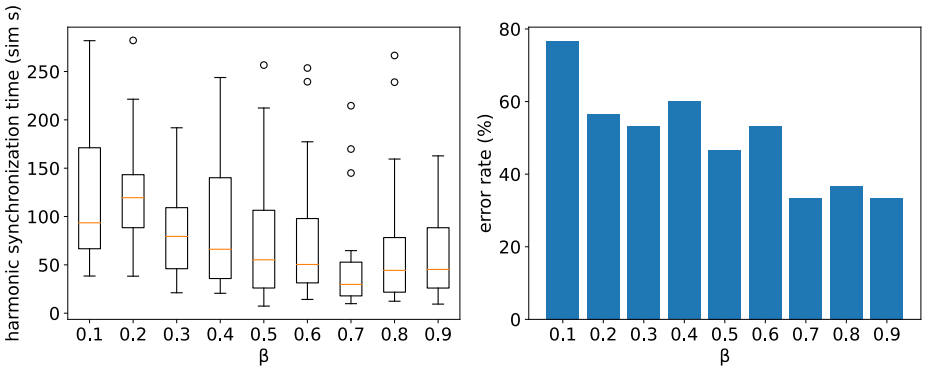


Figure 6.6: Synchronization times (s) for 6 robots with both initially random and unsynchronized phases, and frequencies, for varying frequency coupling constants β. 30 simulation runs per β are reported.

Experiment analysis: Also, from the results in Figure 6.6, it becomes apparent that $\beta = 0.7$ might be the best choice to continue using in our Unity synchrony simulator—at least when it comes to collective sizes $|R| = 6$.

6.3.3 Increasing degree of self awareness

Motivations and intentions are exactly the same as in 6.2.5, although now it is time to explore the effects of various degrees and scopes of self awareness in the second and more challenging phase and frequency ($\phi\&\omega$) synchronization problem.

6.3.3.1 Self awareness scope tuning

Similarly to in the experiments in 6.2.5.1 we here want find out whether global connections in the pulse coupled oscillators really is necessary to maintain performance in harmonic synchronization, but for this time in the second and more complex ($\phi\&\omega$) synchronization problem.

k_s nearest neighbour scope

The first *self awareness scope* scenario is experimented with and tuned. The number of neighbouring Dr. Squiggle robots each individual Dr. Squiggle robot will listen for fire signals to (k_s) will be scaled and increased from $k_s = 1$ up until $k_s = |R| - 1$ which is the maximum number of neighbours each Dr. Squiggle robot can listen to for fire signals.

d_s radial scope

6.3.3.2 Simulation breaking

Experiment setup: In order to test the limits of the developed synchronization simulator, we will here investigate the effects in harmonic synchronization performance when scaling and exploding the musical robot collective size

$|R|$—also given some various self awareness scopes found worthwhile to further investigate in the previous experiments in 6.3.3.

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'.

].

7.1 General discussion

7.2 Conclusion

7.3 Further work

Physical domain: Going from simulation to the real physical world, using M. J. Krzyzaniak and RITMO's actual musical robots, the Dr. Squiggles (cf. 2.7.2). This would involve implementing or translating the software written for the simulated Dr. Squiggles in the Unity simulator environment into the physical hardware and interfaces available in the real Dr. Squiggle robots. Using an audio interface with the Dr. Squiggles robots in order to transmit or play audible "fire" signals would perhaps correspond to *self awareness scope 2* as defined in this thesis, whereas *self awareness scope 3* could possibly correspond in reality to communication technologies where messages are heard globally relatively instantaneously after it is transmitted, such as e.g. wireless local area networks (e.g. Wi-Fi) or—even faster—Ethernet internet cables.

Introduction of noise and epistemolgical challenges: In this music system, pulses (or firings) were designed in such a way that the other musical nodes would easily hear the distinct signals. We could have easily imagined a scenario

where this would not be given; hence calling for more sophisticated methods for detecting and separating between actually separate (spatially) audio-sources. Epistemological explorations, related to e.g. Computational Auditory Scene Analysis (as mentioned in [11]) would then be of interest.

Automatic and online (harmonic) fire sound generation: As alluded to in 5.1.3.2, one possible avenue to explore in order to find harmonious chords or tones—when played together—in a more automatic approach, live and online during simulation, could be found through musical machine learning models predicting fitting and accompanying chords and harmonies (given a tone) in real-time, like the ones tested thoroughly in B. Wallace's interactive music system PSCA [15]. This was however not attempted in this thesis, as this was beyond the scope of the project.

Hindrances throughout the thesis work: The main reason why these aforementioned ideas were not possible to explore throughout this thesis work in the end, was due to an (to me at least) unseen and—for months—unobservable Unity peculiarity (perhaps only to a Unity novice like me) which in turn caused visible bugs in two mechanisms—namely the harmonic synchrony detection mechanism as well as my implementation of K. Nymoen et al.'s frequency adjustment method. So given that I only saw problems in the harmonic synchrony detection and frequency synchronization, I ended up spending considerable time throughout the thesis period trying to debug these two mechanisms, without success—or if any then with more bugs popping up. The way out of this extremely confusing and frustrating situation reared its head eventually as my attention was brought to the synchrony simulator's determinism. Little did I know that once I made my synchrony simulator deterministic, the problems I had been debugging for months were solved immediately. And after discussing with supervisors, and a much more experienced Unity user (Frank Veenstra), the root (and to me unseen) problem, and the reasons why it lead to those visible problems in my harmonic synchrony detection and frequency synchronization, suddenly made total sense. And so now this resulting knowledge about indeterminism in Unity, learnt the hard way, will most likely be shared through a ROBIN wiki page or the likes. Hopefully, this will prevent future UiO robotics master's students from stumbling onto the same problems I did, and from ending up in similar rabbit holes as I ended up in.

Bibliography

[1] Arjun Chandra et al. "Enabling participants to play rhythmic solos within a group via auctions". In: *9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012), Queen Mary University of London* (2012).

[2] Arjun Chandra et al. "Reference Architecture for Self-aware and Self-expressive Computing Systems". In: *Self-aware Computing Systems: An Engineering Approach*. Ed. by Peter R. Lewis et al. Natural Computing Series. Cham: Springer International Publishing, 2016. Chap. 4, pp. 37–49. ISBN: 978-3-319-39675-0. DOI: 10.1007/978-3-319-39675-0_4.

[3] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization". In: *IEEE computational intelligence magazine* 1.4 (2006). Publisher: IEEE, pp. 28–39.

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

[5] Auke Jan Ijspeert. "Central pattern generators for locomotion control in animals and robots: a review". In: *Neural networks* 21.4 (2008). Publisher: Elsevier, pp. 642–653.

[6] Keiji Konishi and Hideki Kokame. "Synchronization of pulse-coupled oscillators with a refractory period and frequency distribution for a wireless sensor network". In: *Chaos: an interdisciplinary journal of nonlinear science* 18.3 (2008). Publisher: American Institute of Physics, p. 033132.

[7] Samuel Kounev et al. "The Notion of Self-aware Computing". In: *Self-Aware Computing Systems*. Ed. by Samuel Kounev et al. Cham: Springer International Publishing, 2017. Chap. 1, pp. 3–16. ISBN: 978-3-319-47474-8. DOI: 10.1007/978-3-319-47474-8_1.

[8] 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.

[9] Peter R. Lewis, Arjun Chandra, and Kyrre Glette. "Self-awareness and Self-expression: Inspiration from Psychology". In: *Self-aware Computing Systems: An Engineering Approach*. Ed. by Peter R. Lewis et al. Natural Computing Series. Cham: Springer International Publishing, 2016.

Chap. 2, pp. 9–21. ISBN: 978-3-319-39675-0. DOI: 10.1007/978-3-319-39675-0_2.

[10] 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.

[11] Kazuhiro Nakadai and Hiroshi G. Okuno. "Robot audition and computational auditory scene analysis". In: *Advanced Intelligent Systems* 2.9 (2020). Publisher: Wiley Online Library, p. 2000050.

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

[13] Anton V. Proskurnikov and Ming Cao. "Synchronization of pulse-coupled oscillators and clocks under minimal connectivity assumptions". In: *IEEE Transactions on Automatic Control* 62.11 (2016). Publisher: IEEE, pp. 5873–5879.

[14] Thomas Schmickl et al. "CoCoRo–The Self-Aware Underwater Swarm". In: *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*. IEEE, 2011, pp. 120–126.

[15] Benedikte Wallace. "Predictive songwriting with concatenative accompaniment". Master's Thesis. 2018.