Real-time Programming and Processing of Music Signals

Arshia Cont

Realtime Programming & Processing of Music Signals

by

ARSHIA CONT

Ircam-CNRS-UPMC Mixed Research Unit
MuTant Team-Project (INRIA)

Musical Representations Team,
Ircam-Centre Pompidou
1 Place Igor Stravinsky, 75004 Paris, France.

Habilitation à diriger la recherche

Defended on May 30th in front of the jury composed of:

Gérard Berry	Collège de France	Professor
Roger Dannanberg	Carnegie Mellon University	Professor
Carlos Agon	UPMC - Ircam	Professor
François Pachet	Sony CSL	Senior Researcher
Miller Puckette	UCSD	Professor
Marco Stroppa		Composer

à Marie

le sel de ma vie

CONTENTS

I

INTRODUCTION

This document is a synthetic summary of my research activity at the intersection of machine learning, signal processing and computer science applied to music. The ultimate goal of this research is to empower computers with musical intelligence capable of interactions with live (human) musicians on stage. This goal is inline with current practices of *mixed music* which is defined as the association of human musicians with computers, both at the time of authoring (music composition) and realtime execution (music performance). To reach this goal, computers must be capable of undertaking *machine listening* in realtime and use this artificial perception to coordinate temporal activities in reaction to the environment. To this end, this manuscript summarizes research in this domain within two main themes of *Machine Listening* and *Reactive Programming*, and studies the coupling and necessary interaction between the two, with the aim of leveraging *Interactive Music Systems* towards *Cyber-Physical Music Systems* for digital art.

1.1 Synthetic Summary

Computer Music as a research discipline deals with many facets of computer science related to audio: signal processing, object recognition, machine learning, computer languages, physical modeling, and perception and cognition, just to name a few. In its practice, it brings in various disciplines with the aim of making music with the aid of (or for) computers. Among long practical traditions in computer music, *Interactive music systems* (Rowe 1992) have gained tremendous attention in both research and artistic communities within the past 15 years. Within this paradigm, the computer is brought into the cycle of musical creation as an *intelligent performer* (Vercoe 1984) and equipped with a *listening machine* (Scheirer 2000) capable of analyzing, coordinating and anticipating its own and other musicians' actions within a musically coherent and synchronous context. Interactive music systems for computer music composition and performance were introduced in mid-1980s at Ircam. Their use have become universal ever since and their practice has not ceased to nourish multi-

disciplinary research. From a research perspective, an interactive music system deals with two problems: realtime machine listening (Rowe 1992; Scheirer 2000) or music information retrieval from musicians on stage, and music programming paradigms (Loy et al. 1985; Assayag and Gerzso 2009) reactive to the artificial listening. Whereas each field has generated subsequent literature, few attempts have been made to address the global problem by putting the two domains in direct interaction. This research directly aims at modeling of such interactions between the two domains by attempting to foster research in each respective field and bringing humans in the loop of sound and music computing.

The first aspect of interactive music systems is the act of *machine listening* or formally speaking *online music information retrieval*. The goal of such systems is to extract low-level information (such as polyphonic pitches, audio descriptors) along with high-level musical information (such as tempo or musical pace, position in a score, formal structures) in realtime. Existing approaches make large use of machine learning and signal processing and proposed models are highly application-dependent and most often *off-line* (meaning the system has access to future data). Realtime machine listening is still a major challenge for artificial sciences that should be addressed both on application and theoretical fronts. We present our efforts on this topic in Chapter 2. In Sections 2.1 and 2.2 we present two state-of-the-art methods in terms of modeling and performance for two complex tasks of realtime transcription and alignment of polyphonic audio signals. In Section 2.3, we present our work in formulating a novel mathematical framework for statistical representation of audio streams called *Music Information Geometry*, that attempts to address fundamental problems in high-level Music Information Retrieval (MIR) tasks for quantification and qualification of information content. This work leverages traditional signal representations with tools from information theory and differential geometry, to provide mathematical concepts that in return will simplify (often complex) problem formulations in MIR tasks such as automatic segmentation (Section 2.3.2) and automatic structure discovery (Section 2.3.3).

Chapter 3 presents our efforts on reactive synchronous programming in computer music, or simply the formalization of reactions in interactive systems to artificial perceptions discussed previously. Our approach is motivated by current practices in digital arts for *Authoring* or programming of pieces, and their *realtime performances* deterministic with regards to authored patterns and non-deterministic with regards to interpretation parameters. Section 3.1 reviews such practices and studies common software architecture to this end. To address several scientific and practical issues brought by this study, we present Antescofo in section 3.2, a performance-synchronous language for computer music. The system and its language, running publicly since 2009, has now established itself worldwide for mixed music repertoire. Highly inspired by synchronous-reactive languages (Halbwachs 1993), we informally describe the language in section 3.2.1 and study its runtime semantics in section 3.2.2.

Chapter 4 presents our recent and future efforts in bringing the two topics discussed within a unique theoretical and practical framework. We aim to leverage the traditional notion of *Interactive Music Systems* to *Cyber-Physical Music Systems (CPS)*, by showing the differences in their design process and possible

gains when considering the latter both in theory and practice. Specifically, we revisit common practical problems in computer music such as the community divisions as a result of domain-specific software and languages, important issues in realtime computing and nomadic and distributed deployment in Section 4.2. The scientific context of this prospective research is presented in Section 4.3, drawn from CPS literature (Kim et al. 2012; E. A. Lee and Seshia 2011), exposing our plans for establishing a literature on Models of Computations, strong temporal semantics, and architectures for embedded computer auditions with important impact on digital art practices and issues such as preservation of digital art programs.

1.2 Publication List 2007-2012

Journals

[1] Cont, Arshia (May 2010). "A coupled duration-focused architecture for realtime music to score alignment." In: 32.6, pp. 974–987. URL: http://articles.ircam.fr/textes/Cont09a/.

[2] — (May 2012b). "Modélisation anticipative des systèmes musicaux. Reconnaissance, génération, synchronisation et programmation synchrone temps réel en informatique musicale." French. In: 31.3, pp. 311–335. DOI: 10.3166/tsi.31.311-335. URL: http://hal.inria.fr/hal-00699290.

[3] — (May 2012c). "Synchronisme musicale et musiques mixtes: Du temps écrit au temps produit." French. In: 22.1, pp. 9–24. URL: http://hal.inria.fr/hal-00698922/PDF/Cont_eI_preuves2_22-1.pdf.

[4] Cont, Arshia, Shlomo Dubnov, and Gerard Assayag (May 2011). "On the Information Geometry of Audio Streams with Applications to Similarity Computing." In: 19.4. URL: http://articles.ircam.fr/textes/Cont10b/index.pdf.

[5] Dessein, Arnaud and Arshia Cont (April 2013). "An information-geometric approach to real-time audio segmentation." In: 20.4, pp. 331–334. DOI: 10.1109/LSP.2013.2247039. URL: http://hal.inria.fr/hal-00793999.

Popular Science Articles

[6] Cont, Arshia (July 2011a). "Interaction musicale en temps réel entre musiciens et ordinateur : L'informatique musicale, de l'accompagnement musical automatique vers la programmation synchrone en temps réel." In: URL: http://interstices.info/interaction-musicale (visited on 08/11/2012).

[7] — (June 2012a). "L'ordinateur qui joue comme un musicien." In: 465, pp. 68–72.

[8] Cont, Arshia, Florent Jacquemard, and Pierre-Olivier Gaumin (November 2012). "Antescofo à l'avant-garde de l'informatique musicale." In: URL: http://interstices.info/antescofo (visited on 02/01/2013).

Book Chapters

[9] Cont, Arshia, Gerard Assayag, et al. (August 2010). "Interaction with Machine Improvisation." In: *The Structure of Style*. Ed. by Shlomo Dubnov Kevin Burns Shlomo Argamon. Springer Berlin Heidelberg, pp. 219–246. ISBN: 978-3-642-12337-5. URL: http://dx.doi.org/10.1007/978-3-642-12337-5_10.

[10] Cont, Arshia, Shlomo Dubnov, and Gerard Assayag (2007a). "Anticipatory Model of Musical Style Imitation using Collaborative and Competitive Reinforcement Learning." In: *Anticipatory Behavior in Adaptive Learning Systems*. Ed. by Butz M.V. et al. Vol. 4520. Lecture Notes in Computer Science / Artificial Intelligence (LNAI). Springer Verlag, pp. 285–306. ISBN: 978-3-540-74261-6. URL: http://www.springerlink.com/content/978-3-540-74261-6/.

[11] Dessein, Arnaud, Arshia Cont, and Guillaume Lemaitre (2013). "Real-Time Detection of Overlapping Sound Events with Non-Negative Matrix Factorization." In: *Matrix Information Geometry*. Ed. by Frank Nielsen and Rajendra Bhatia. Springer Berlin Heidelberg, pp. 341–371. ISBN: 978-3-642-30232-9. URL: http://dx.doi.org/10.1007/978-3-642-30232-9_14.

International Conferences

[12] Cont, Arshia (August 2008a). "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." In: *Proceedings of International Computer Music Conference (ICMC)*. Belfast. URL: http://articles.ircam.fr/textes/Cont08a/.

[13] — (July 2011b). "On the creative use of score following and its impact on research." In: *Sound and Music Computing*. URL: http://articles.ircam.fr/textes/Cont11a/.

[14] Cont, Arshia, Shlomo Dubnov, and Gerard Assayag (September 2007b). "GUIDAGE: A Fast Audio Query Guided Assemblage." In: *Proceedings of International Computer Music Conference (ICMC)*. ICMC Best Presentation Award. Copenhagen.

[15] Cont, Arshia, Shlomo Dubnov, and David Wessel (September 2007). "Realtime Multiple-pitch and Multiple-instrument Recognition For Music Signals using Sparse Non-negative Constraints." In: *Proceedings of Digital Audio Effects Conference (DAFx)*. Bordeaux.

[16] Cont, Arshia, José Echeveste, et al. (September 2012). "Correct Automatic Accompaniment Despite Machine Listening or Human Errors in Antescofo." In: *International Computer Music Conference (ICMC)*. URL: http://hal.inria.fr/hal-00718854/PDF/Antescofo_ICMC2012_new.pdf.

[17] Cont, Arshia, Jean-Louis Giavitto, and Florent Jacquemard (2013). "From Authored to Produced Time in Computer-Musician Interactions." In: *CHI 2013 Workshop on Avec le Temps! Time, Tempo, and Turns in Human-*

Computer Interaction. John Thomas, Yue Pan, Thomas Erickson, Eli Blevis, Catherine Letondal, Aurélien Tabard. ACM. URL: http://hal.inria.fr/hal-00787033.

[18] Cont, Arshia, Diemo Schwarz, Norbert Schnell, and Christopher Raphael (September 2007). "Evaluation of Real-Time Audio-to-Score Alignment." In: *International Symposium on Music Information Retrieval (ISMIR)*. Vienna, Austria.

[19] Dessein, Arnaud, Arshia Cont, and Guillaume Lemaitre (August 2010). "Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence." In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*.

[20] Dubnov, Shlomo, Gerard Assayag, and Arshia Cont (September 2007). "Audio Oracle: A New Algorithm for Fast Learning of Audio Structures." In: *Proceedings of International Computer Music Conference (ICMC)*. Copenhagen.

[21] Dubnov, Shlomo, Arshia Cont, and Gérard Assayag (September 2011). "Audio Oracle Analysis of Musical Information Rate." In: *Proceedings of IEEE Semantic Computing Conference (ICSC2011)*, pp. 567–571. URL: http://articles.ircam.fr/textes/Dubnov11a/.

[22] Montecchio, Nicola and Arshia Cont (May 2011a). "A Unified Approach to Real Time Audio-to-Score and Audio-to-Audio Alignment Using Sequential Montecarlo Inference Techniques." In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[23] — (September 2011b). "Accelerating the Mixing Phase in Studio Recording Productions by Automatic Audio Alignemtn." In: *12th International Symposium on Music Information Retrieval (ISMIR)*. in press.

[24] Nouno, Gilbert et al. (July 2009). "Making an orchestra speak." In: *Sound and Music Computing*. SMC2009 Best Paper Award. URL: http://articles.ircam.fr/textes/Nouno09a/.

1.3 Research Advising Summary

PhD Thesis Advising

[1] Dessein, Arnaud (December 2012). *Computational Methods of Information Geometry with Real-Time Applications in Audio Signal Processing*. URL: http://hal.inria.fr/tel-00768524/PDF/Dessein2012PhD.pdf.

[2] Echeveste, José (Travaux démarés en 2011). *Real-time Synchronous language for authoring of musical interaction*.

[3] Montecchio, Nicola (April 2012). *Alignment and Identification of Multimedia Data: Application to Music and Gesture Processing*. (Encadrement à 25%). URL: https://sites.google.com/site/nicolamontecchio/montecchio%20-%20phd%20thesis.pdf?attredirects=0.

Master Thesis Advising

[4] Baudart, Guillaume (September 2012). *Antescofo: Vers une programmation synchrone.*

[5] Cauchi, Benjamin (2011). *Non-Negative Matrix Factorization Applied to Auditory Scenes Classification.* Master. URL: http://articles.ircam.fr/textes/Cauchi11a/index.pdf.

[6] Cuvillier, Philippe (September 2012). *Suivi de partition: étude du cadre multi-objets pour l'inférence de position.*

[7] Dessein, Arnaud (2009). *Incremental multi-source recognition with non-negative matrix factorization.* Master. URL: http://articles.ircam.fr/textes/Dessein09b/index.pdf.

[8] Echeveste, José (2011). *Stratégies de synchronisation et gestion des variables pour l'accompagnement musical automatique.* Master. URL: http://articles.ircam.fr/textes/Echeveste11a/index.pdf.

[9] Franchon, Lea (2012). *Temporal Analysis of Mixed Intrumental/Electronic Music Scores.*

[10] Ricordeau, Olivier (2006). *Sélection de descripteurs pour modèles d'observation audio temps-réel.* Master.

PhD Defense Committees

[11] Hennequin, Romain (November 2011). *Decomposition of musical spectrograms informed by spectral synthesis models.* URL: http://www.romain-hennequin.fr/doc/these.pdf.

[12] Joder, Cyril (September 2011). *Alignement temporel musique-sur-partition par modèles graphiques discriminatifs.* URL: http://www.mmk.ei.tum.de/~jod/publis/Joder_These.pdf.

[13] Lefèvre, Augustin (September 2012). *Dictionary learning methods for single-channel audio source separation.*

[14] Toro, Mauricio (September 2012). *Structured Interactive Scores.*

II REALTIME MACHINE LISTENING

In interactive music systems, Machine Listening defines the dynamics of the physical environment and provides the necessary artificial perception to enable (timely) reactions from computer components. Machine listening consists of realtime extraction of low-level and high-level sound and music parameters from audio streams.

The first thorough study of machine listening appeared in Eric Scheirer's PhD thesis at MIT Media Lab (Scheirer 2000) with a focus on low-level listening such as pitch and musical tempo, paving the way for a decade of research. Since the work of Scheirer, the literature has focused on task-dependent methods for machine listening such as pitch estimation, beat detection, structure discovery and more. Unfortunately, the majority of existing approaches are designed for information retrieval on large databases or off-line methods. Whereas the very act of listening is *realtime*, very little literature exists for supporting realtime machine listening.

In this chapter we present research works on realtime machine listening. The common factor of all presented applications and framework is their online and realtime nature on incoming audio streams into a system.

2.1 Automatic Transcription

Automatic transcription refers to the mapping of audio signals to symbolic representations of music such as notes (pitches), instrument types and rhythms in western music. Among different variants of automatic transcription problems that of multiple-pitch (or multiple-f_0) estimation in the presence of multiple sources of sounds is of particular difficulty and interest. This problem has been widely studied in the literature employing various approaches in signal processing and machine learning (D.-L. Wang et al. 2006). While most methods have focused on off-line approaches with complex signal processing, since 2006 we have focused on on-line and realtime approaches using relatively simple machine learning techniques that learn and adapt to the environment complexity. This section is a summary of works in (Cont 2006b; Cont, Dubnov, and Wes-

sel 2007; Dessein 2009; Dessein et al. 2010; Dessein et al. 2013) on automatic transcription.

In 2006, we were among the first to show that the hard problem of multiple-pitch detection can be solved in realtime and employing non-negative matrix factorization (NMF) techniques. The original NMF framework as proposed in (D. D. Lee et al. 2001) aims at factorizing an input $n \times m$ matrix $\mathbf{V}$ into an $n \times r$ matrix $\mathbf{W}$ and an $r \times m$ matrix $\mathbf{H}$ such that $\mathbf{V} \approx \mathbf{WH}$ and where all three matrices are non-negative. In case of sound, non-negative multivariate short-term representations of sound are stacked into $\mathbf{V}$ where each column could then be expressed as $\mathbf{v}_j \approx \mathbf{W}\mathbf{h}_\mathbf{j} = \sum_i h_{ij}\mathbf{w}_i$. The columns of $\mathbf{W}$ thus form a *basis* whereas the rows in $\mathbf{H}$ represent the *evolution* of each basis through columns of $\mathbf{V}$ (or time in case of sound). The aim of the NMF algorithm is thus to improve the approximate factorization by optimizing a given goodness-of-fit measure called *cost function* $\mathcal{C}(\mathbf{V}, \mathbf{WH})$ over both $\mathbf{W}$ and $\mathbf{H}$. The standard formulation in (D. D. Lee et al. 2001) employs a Frobenius norm for $\mathcal{C}$ through iterative multiplicative updates using a gradient descent scheme with proof of convergence. A flourishing literature exists on variants and extensions of the standard NMF in terms of modified models (e.g. using tensors), modified constraints (e.g. employing sparsity), and of modified cost functions and regulations (Cichocki et al. 2009).

Our proposed framework for realtime pitch detection using NMF debuted in (Cont 2006b) by employing a short-time spectral amplitude representation based on instantaneous fixed-point analysis of realtime audio streams (Kawahara et al. 1999) for the frontend representation.The basis matrix $\mathbf{W}$ contains previously learned templates of all available pitches (e.g. all 88 keys on a Piano) equipped with a noise template. $\mathbf{V}$ would represent the realtime vector of incoming audio in the aforementioned representational scheme. Realtime multiple-pitch detection then boils down to finding the optimal $\mathbf{H}$ or

$$\underset{\mathbf{H} \in \mathbb{R}_+^{r \times m}}{\arg\min} \quad \mathcal{C}(\mathbf{V}, \mathbf{WH}) \tag{2.1}$$

where $\mathbf{W}$ is constant. In the absence of future information or complete matrix representation in $\mathbf{V}$ the application of standard NMF would quickly become limited (e.g. off-line transcriptions in (Abdallah et al. 2004; Smaragdis et al. 2003)). Additivity and non-negativity are no longer sufficient constraints to obtain satisfactory results. One among many reasons for this is the correlation and inter-relations between harmonic templates in $\mathbf{W}$ such as octave relations that would lead to multiple local optimal results for a given $\mathbf{v}$. To overcome this issue, and following (Hoyer 2004) we proposed a gradient-descent upgrade for eq. 2.1 with sparsity constraints on $\mathbf{H}$ as a combination of ℓ_ϵ and ℓ_2 norms (Cont 2006b). For each realtime audio input, the system would undertake an iterative procedure by geometrically solving the intersections of ℓ_ϵ and ℓ_2 hyper-planes on the gradient-descent steps and using a Kullback-Leibler divergence as cost function. The proposed system was capable of decomposing realtime audio into pre-learned pitch templates and a realtime implementation has been in use since then by various artists and composers.

Following the experiment and results obtained for multiple-pitch estimation in (Cont 2006b), we attempted to extend the proposed techniques to more complex situations such as realtime decoding of multiple-pitches and multiple-instruments in one shot and at the same time. Our paper in (Cont, Dubnov, and Wessel 2007) proposes this framework by adapting the representational frontend in (Cont 2006b) to *modulation spectrum* (Sukittanon et al. 2004) which had previously shown evidence in disciminating both pitches and instrument types on audio signals (Dubnov et al. 2003). Besides being the only realtime system capable of decoding both pitches and instruments, results in (Cont, Dubnov, and Wessel 2007) can compete with state-of-the-art systems for instrument classification.

Besides interesting performance, the experiments in (Cont, Dubnov, and Wessel 2007) showed that the previous optimization process in (Cont 2006b) using linear gradient-descent and geometric sparsity controls were insufficient for more complex geometric domain such as the one used for instrument classification. More powerful optimization schemes had to be integrated to address such problems leading to the Masters thesis of Arnaud Dessein with the proposition of convex quadratic programming (CQP) and second-order cone programming instead of mere gradient descent updates (Dessein 2009). Within this framework sparsity controls are no longer parameters (as in (Cont 2006b)) but random variables solved by the system itself. This led to the following optimization scheme,

$$\underset{\mathbf{H}\in\mathbb{R}_+}{\arg\min} \quad \frac{1}{2}\|\mathbf{V}-\mathbf{W}\mathbf{H}\|_2^2 + \lambda_1\|\mathbf{H}\|_1 + \frac{\lambda_2}{2}\|\mathbf{H}\|_2^2 \tag{2.2}$$

where the ℓ_1-norm penalizes less sparse vectors, and the ℓ_2-norm is a particular case of Tikhonov regularization that under certain conditions renders the optimization problem strictly convex. The above equation lead to tractable multiplicative updates (See Dessein 2009, Section 2.3.2).

Extensions from sparse NMF to CQP allowed better control over algorithm parameters as well as considered more complex problems of event detection beyond traditional multiple-pitch estimation. Noticeably, we were able to use other types of templates as dictionary elements in $\mathbf{W}$ such as non-harmonic drum and percussion sounds as well as quasi-stationary environmental sounds with applications to complex computational auditory scene analysis as shown in (Dessein et al. 2013).

Our realtime multiple-pitch detection algorithm using these methods was ranked 2nd in the international *Music Information Retrieval Evaluation Campagne (MIREX)* being the only realtime system present (See MIREX 2010). The proposed methods can be also used for realtime detection of events in complex auditory scenes where a dictionary of expected events is available. We have recently put our efforts together with the *Perception and Sound Design Team* at Ircam on this subject to apply these methods to real auditory scene analysis such as train station soundscapes (Cauchi et al. 2012). We will continue strengthening both the algorithmic and applicative aspects of this approach in the four years to come.

2.2 Automatic Alignment

Automatic alignment refers to automatic ordering of two or more signals representing the same approximate content but with different temporal structure and most often with different nature so that there is a one-to-one correspondence between them. Alignment techniques are often used as frontends for many Music Information Retrieval techniques and applications. For example, it is important for many such applications to obtain a one-to-one correspondence between an audio performance and its symbolic score, or two audio performances of the same piece, or between control signals such as gestures. When the task of automatic alignment is undertaken in realtime it is often referred to as *Score Following*. Robust score following is a difficult task due to the variability of performance parameters among human musicians, the difficulty of handling clutters and errors in the environment and polyphonic and complex nature of audio scenes in question. Moreover, score followers are first class citizens in interactive and mixed music repertoire since their inception in the 1980s (Dannenberg 1984; Vercoe 1984) thanks to artists such as Pierre Boulez and Philippe Manoury with their early adoption of this technology in their live electronic pieces (Manoury 1990; Manoury 2007; Szendy 1998). Alignment and score following techniques developed by this author constitute the state-of-the-art in terms of modeling and performance up to this date. This section summarizes current and future research on this topic from (Cont 2006a; Cont, Schwarz, Schnell, and Raphael 2007; Cont 2010; Montecchio et al. 2011a; Cont 2011; Montecchio et al. 2011b).

An important turning point in realtime modeling for score follower are papers (Cont 2010; Cont 2008a) which mark the beginning of the Antescofo system. Before these works, state-of-the-art score following systems were dominated by probabilistic graph networks and Hidden Markov Models (HMMs) (Raphael 1999; Raphael 2001; Orio et al. 2001) as adoptions of their equivalent models in speech recognition. In these methods, target symbolic scores are employed to produced Markovian structures that are inversely assumed to generate incoming audio to obtain a generative probabilistic framework for decoding of hidden states during a realtime performance. As an illustrative example, Figure 2.1 shows a typical Markov model used for a single music event (note, chords, etc.) in such approaches. The inverse and generative assumptions of these models represent the following major bottlenecks for modeling of audio streams and music performances:

1. State occupancy or duration models are implicit (and a function of r, p and q in Fig. 2.1);

2. State-space models representing scores remain static after construction whereas a music performance unlike speech can be extremely dynamic (structure and parameters in Fig. 2.1 remain static during decoding);

3. Observation likelihood models (or the probability of being in a given state s_i given audio stream y, $P(y|s_i)$) represent a major aspect of modeling despite extreme uncertainties within the acoustic environment especially in polyphonic and noisy situations; and

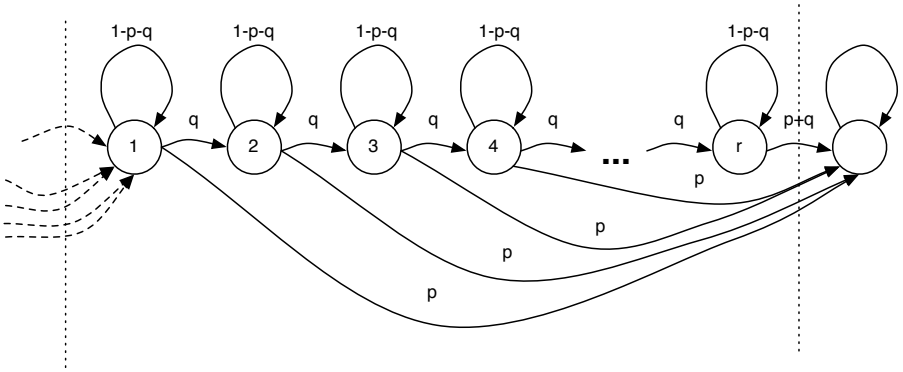


Figure 2.1: Typical Markovian structure of a music event (See Cont 2010, for details).

4. The Markov assumption for state transitions does not seem to hold at both macro and micro level of music information whereas in traditional HMMs for sound and speech it is assumed allover.

Besides modeling and theoretical issues (with direct impact on performance), such systems do not show the needed tolerance during realtime performance in real musical situations and usually require offline training on performance data (such as music rehearsals) (Cont, Schwarz, and Schnell 2004; Raphael 2001). After several experiments with such systems in (Cont, Schwarz, and Schnell 2004; Cont, Schwarz, and Schnell 2005; Cont 2004) it became apparent that fundamental reconsideration of the assumed models were necessary. Our first attempt was the use of Hierarchical HMMs with Particle Filtering in (Cont 2006a) which to some extents addressed items (3) and (4) especially for polyphonic signals but still showed extreme complexity and non-intractability of model parameters, particularly in circumstances of poor performance. It became clear that new foundations for modeling of time in audio stream analysis was a necessity which were brought into life within two fronts:

2.2.1 Discrete State-space Models of Time

In 2007 and in collaboration with composer Marco Stroppa, new foundations for modeling of music events in view of realtime alignment were set. The proposed model is based on a state-space generative model of audio with the following premises:

1. Markovian assumption only holds in transition between macro events. The structure once inside an event is *not* Markovian or at best is a dynamic variable-order Markov structure.

2. Durations, sojourn and survival times for events are first-class citizens of the model and not the result of decoding. This leads to *explicit* and *duration-focused* models of events that are dynamic during performance.

3. Observation likelihood on a single source of information (such as spectral pitch or structure) can be fallible even in best-cases of real acoustic sit-

uations. Combinations of several sources of information, even fallible, is stronger than one.

4. Model parameters are dynamic and adaptive. Parameter learning is *active* and online and a function of environment behavior.

The above considerations led to the model presented in (Cont 2010) which consists of *hybrid* states representing both *Markov* and *Semi-Markov* states within a unified inference framework. Semi-Markov states have explicit sojourn and duration probabilities modeled as a Poisson Process on a circular phase domain where one cycle represents a musical beat. A macro-event (such as notes, chords, trills etc.) that occupy both space and time are represented as a *single* semi-markov state whereas atemporal events (such as grace notes or non-indicated silences) are represented as Markov states (with implicit duration). Figure 2.2 shows a sample score using such modeling. Audio observation module is ex-

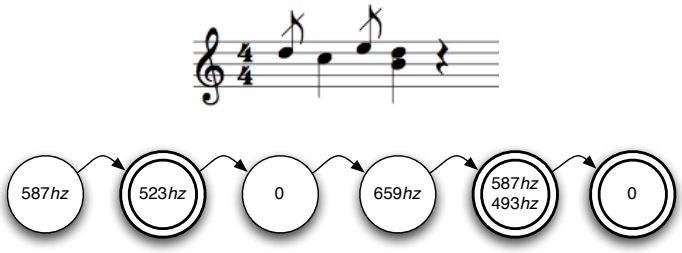


Figure 2.2: A sample hybrid state-space structure using Semi-Markov (double lined) and Markov (regular) states (See Cont 2010, for details).

tremely simple and computationally cheap and complemented by a time-agent modeled as a linear Kalman-Filter in the circular domain following (Large et al. 1999). Model parameters are being updated in realtime upon each macro-event detection and as a function of realtime performance thus evading the necessity of offline learning. Despite cheap signal processing, the system is capable of following and decoding of tempo and position on highly polyphonic and noisy signals. The proposed method thus lies upon hybrid discrete state-spaces with heterogeneous notions of time (discrete, continuous, relative, absolute).

This method is currently the model used for score following in Antescofo.

2.2.2 Continuous State-space Models of Time

Another idea in improving temporal modeling is to abandon discrete state-space models and switch to the continuous world. Temporal continuity can be achieved either in signal representation (difficult in the case of digital audio), in the state-space modeling, or in the inference scheme. Continuous HMMs are usually intractable in the inference phase leading to approximations. A promising alternative is the use of sequential Montecarlo inference techniques best known as Particle Filtering methods (Arulampalam et al. 2002). These methods were previously explored by the author in (Cont 2006a) for hypothesis testing in uncertain environments. The idea is to take advantage of their

stochastic behavior to represent "time" as a continuous line instead of discrete states. The proposed model in (Montecchio et al. 2011a) allowed not only a continuous representation of time in decoding and inference, but also a unified framework for both realtime audio-to-audio and audio-to-score alignment. Figure 2.3 shows the basic premise of this representation, leaving the details in (Montecchio et al. 2011a).

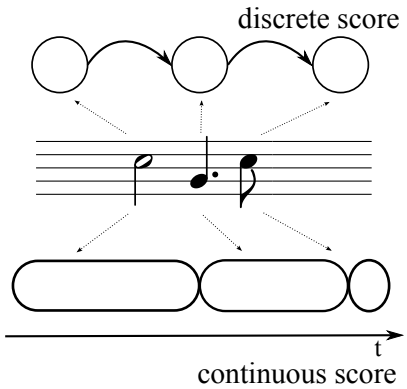


Figure 2.3: Continuous state-space model of a sample score. (See Montecchio et al. 2011a, for details).

The proposed framework in (Montecchio et al. 2011a), due to its strong stochastic nature based on hypothesis testing instead of path optimization in standard HMMs, allows novel applications of automatic alignment such with uniform prior over a score and partial result handling. One such application is proposed in (Montecchio et al. 2011b).

Despite the attractiveness of Particle Filtering methods demonstrated in (Montecchio et al. 2011b; Montecchio et al. 2011a), their employment in realtime systems is still immature and subject to studies by our team among others. As uncertainty grows, the number of particles needed for decoding grows as well leading to more and probably unnecessary computation. Moreover, most existing Particle Filtering approaches use the particles' centroid for the final decoding. This last-stage averaging can cause severe problems in realtime decoding especially in highly uncertain situations. A more rigorous approach is to study the inherent geometry of the stochastic models and undertake geometric walks and manifold calculations on particles instead of mere averaging as proposed in (Snoussi et al. 2009).

In 2006, the author gathered major researchers in the field of automatic alignment and initiated an international campaign for evaluation of such applications. This led to a proposal for "Score Following Evaluation" in MIREX (MIREX 2006) that continues to this date and used worldwide (Cont, Schwarz, Schnell, and Raphael 2007). This international campaign has been running up to this year and will hopefully continue to do so. None of the new participants has yet been able to surpass our results obtained in 2006.

Extensive use of the algorithms in (Cont 2010) implemented in Antescofo not only increased the robustness of the algorithms and shed light on new improvements and additions such as echo cancelations and more, but also brought

interesting challenges and horizons requiring new models and literatures to create and explore. One such problems that will occupy us in the following years is the problem of *asynchrony* and ensemble following (Grubb et al. 1994). In most music ensembles, despite the synchrony of different sound sources (instruments or voices) in the reference, they arrive asynchronously in realtime. Such asynchronies are not only mere artifacts but also a great source of musical interpretation. An idea in decoding such asynchronies is to track several voices within a single (audio) observation. Similar paradigms have been studies over the years in the missile-tracking literature (Mahler 2007). More interestingly these approaches propose explicit models for *clutters* (or false observations) that are difficult to model in generative approaches. Another anticipated advantage of these models is their ability to *focus* on interest regions in the observation instead of considering the entire observation as generated by the model. These models are currently under study by the author and one of his Masters students (See Cuvillier 2012) and will hopefully continue as a PhD project in 2013. Following this research, we hope to be the first to uncover the difficult problem of tracking multiple sources in realtime from a single observation such as music ensemble with various applications to other fields.

2.3 Music Information Geometry

The literature on Information Geometry (Amari et al. 2000) is an emerging field of applied mathematics that aims at studying the notions of statistical inference by using concepts from differential geometry. The main idea is that most parametric families of probability distributions in wide use in engineering applications, can be equipped with an intrinsic geometrical structure of differential manifold. Studying statistical inference in such structures ensures that the results of inference are invariant under arbitrary choice of a parametrization, which is major goal in modeling. Moreover, it creates natural analogies between statistical notions and geometrical interpretations related to notions of differential geometry. Methods of information geometry have been successfully applied to image analysis (Nielsen 2009), Computational Anatomy (Pennec 2009), Radar Processing (Barbaresco 2009), and tracking (Snoussi et al. 2009). Since 2009, we aimed at leading this research in the field of realtime audio processing and by playing a major role in the general theoretical literature. Early results are reported in (Cont, Dubnov, and Assayag 2011; Cont 2008b) following Arnaud Dessein's PhD on this topic is covering this topics for Audio Streams (Dessein 2012).

2.3.1 Motivations and Approach

Music Information Retrieval (MIR) systems deal one way or another with the information content of music signals, their transformations, or extraction of models or parameters from this information. A common question that many such systems ask at their frontend is what information is presented in the signal and to what relevancy? This question is central in almost all music information retrieval systems dealing either with temporal structures of audio data streams

for search applications (query-by-humming, audio matching, music summarization etc.), or with temporal decomposition of audio (source separation, multiple-source identification, etc.).

The main motivation in studying audio streams using information geometry comes from the fact that most current MIR systems are theoretically dominated by statistical models of audio and music and employing statistical inference to infer low-level or high-level symbolic information out of audio streams. Inherent in almost all MIR systems is the notion of *self-similarity measures* between audio entities as a basis to compare and deduce music structures (Foote 1997); enhanced with geometric concepts in machine learning for building classifiers in supervised problems (genre classification, artist identification, query by example etc.) or clustering data in unsupervised settings (audio search engines, structure discovery etc.). Implicit in these considerations is the construction of a *metric space* employing such similarity measures to deduce equivalence classes. Despite the importance of this consideration, almost no MIR system is based on well-defined metric structure. This drawback is tackled with the mere fact that most widely used pattern matching systems employ *bag of features* models, where audio data is represented to the system *with no temporal order* or with crude approximations on the temporal morphology of audio streams. The temporal order and morphology of music signals is probably one of the most precious sources of information that is unfortunately underestimated in most existing approaches and applications.

We believe that the correct answer to this confusing situation is to tackle the very fundamental aspect of information retrieval which is its correct representation with well-behaved geometric properties. If the very goal of such systems is to compare and deduce similar structures with inherent metric spaces and by preserving the temporal morphologies, then the issue must be addressed at the utmost formalization and frontend of any algorithmic approach.

Most practical uses of Information Manifolds take a bottom-up design approach by first defining the inner product over the statistical structures, and a canonical affine connection to constitute an information geometry; and its leading geometrical constructs such as geodesics, volumes, divergences etc. (e.g. (Pennec 2009)). Our approach is rather a top-down design process where the geometry is automatically induced by a Divergence, which by itself is bijected by an engineering choice of statistical structures over data points (audio analysis frames for example). Recent works by Zhang shows that such top-down approach is possible for many types of divergence functions which are in common use in engineering applications (Zhang 2004). Furthermore, Banerjee et al. have proven the bijection between the canonical family of exponential families and that of Bregman divergences (Banerjee et al. 2005). In our work, we translate this design process onto the daily process of MIR system design, and provide sufficient conditions and definitions within which such divergences can provide metric spaces (Cont, Dubnov, and Assayag 2011). This work, for now, focuses on parametric models employing the general family of exponential distributions and their inherent geometries that are widely used in engineering applications.

The proposed top-down approach is consistent for adoption and adaptation of most engineering approaches to pattern recognition: Many engineering solu-

tions start by assuming a generative model on the data (often belonging to the generic family of exponential distributions). Our current approach paves the way for adoption and eventually better addressing of many existing problems and their solutions. In the following sections we present several results and applications of this framework from (Cont, Dubnov, and Assayag 2011; Dessein 2012; Dessein and Cont 2011), emphasizing that this section constitute an ongoing work:

2.3.2 Automatic Change Detection

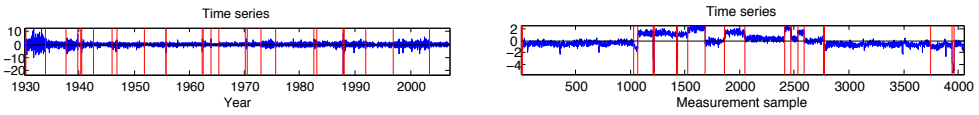
The first step in our modeling of audio streams on information manifolds consists of segmenting an audio stream into *continuous quasi-stationary* chunks that will then be represented as statistical models representing a geometrical object on a Riemannian manifold equipped with the Fisher Information distance. To this end, each incoming audio frame into our system is treated as a *statistical point*, and the problem of segmentation is considered as detecting change points over models that represent a set of points. Within this approach we consider audio information quantity as a *cadlàg* (right continuous with left limits) time-series. Each *model* is thus a parametric statistical distribution formed around a certain parameter vector of that family. We show in (Cont, Dubnov, and Assayag 2011) that this is analogous to choosing a minimum enclosing information ball on statistical points representing the statistical family.

The problem of change detection has been widely studied in the literature within various application contexts (Basseville et al. 1993). The drawback of most approaches is that (1) they are non-causal and off-line; and (2) are based on statistical hypothesis tests where parameters before and after change-point are known (which is not the case in real-life). The most known approach is the *CuSUM* algorithm using Likelihood-Ratio tests on two hypothesis on change and no-change on a signal frame (See original proposal by Page 1954; and further variants by Basseville et al. 1993). Our initial experiments in (Cont, Dubnov, and Assayag 2011) with classical *CuSUM* algorithm in the context of realtime change detection on audio signals further revealed its limitations in practice and a handicap for further progress in our proposal.

Following this we developed online and incremental methods for detecting changes on exponential distribution manifolds by considering model parameters before and after hypothetic change as random variables in the model and integrated within the same Likelihood-Ratio test. Methods of information geometry simply tie maximum likelihood estimation of unknown parameters with their generalized likelihood ratio tests within one single framework and thanks to the inherent geometric duality of expectation and natural parameters (See Dessein 2012).

Figure 2.4 shows change detection results on two datasets commonly used in the literature from financial data (fig. 2.4a) and geological information (fig. 2.4b). Change detection here is obtained by assuming univariate normal distributions over the data allowing changes both in mean and variance (as random variables in the system). Our obtained results conform to the literature (see Dessein 2012, for analysis) and furthermore the system is capable of reporting estimated model

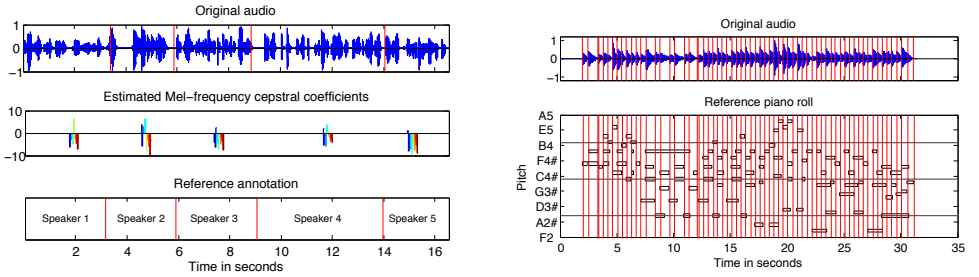
parameters for each segment in the data.



(a) Daily log-return from Dow Jones (b) Well-log data

Figure 2.4: Change detection on real-world datasets

The univariate data in figure 2.4 is univariate but dynamic. The proposed method can also be employed using more complex data as long as the representational frontend for statistical modeling is from the generic family of exponential distributions. Figure 2.5 shows results obtained for online change detection on speech (fig. 2.5a) and polyphonic music (fig. 2.5b). For speech speaker segmentation 12 Mel-frequency cepstral coefficients are used through spherical normal distributions with fixed variance. The top plot in figure 2.5a shows the computed segmentation on the audio waveform, while the bottom plot shows the ground-truth segmentation with the estimated model coefficients (useful for further processing) in the middle. In Figure 2.5b, the goal is to segment the music stream into quasi-stationary chunks which correspond here to music note/chord/event inter-onsets due to the frontend representation employed. For this example the magnitude spectra of audio frames were used to represent frequency histograms naturally modeled as multinomial distributions. The top figure shows the segmentation results on the waveform whereas the lower figure overlaps the segmentation on top of a symbolic transcription of the same audio for comparison.



(a) Speaker segmentation on speech data (b) Note/chord segmentation on polyphonic music data

Figure 2.5: Change detection results on audio and speech

These results demonstrate the capability of this change detection algorithm to approach real-world data with expected results. It should be noted however that the data presented here are of heterogeneous nature, segmentation is done online and the system is *blind* to the nature of the signal in question. The segmented time-series as well as their segment coordinates on the inherent geometry can then be used for further processing.

2.3.3 Automatic Structure Discovery

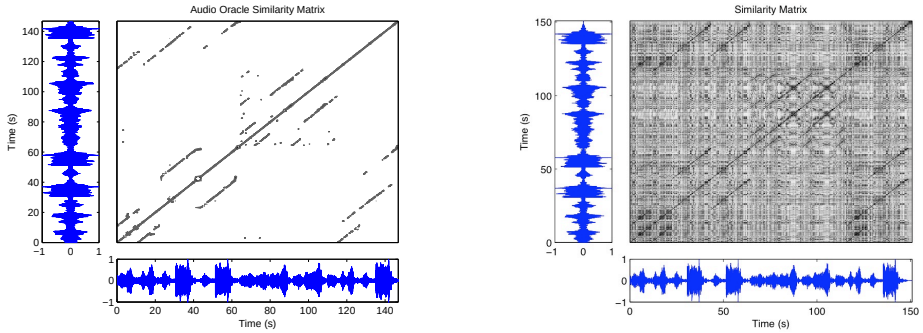
Automatic structure discovery is the problem of automatically detecting recurring structures in an audio stream revealing formal structures in music and is useful for many applications such as audio thumb nailing, fingerprinting and compression. For music structures, whereas the problem is trivial to an untrained human listener, despite all sorts of possible variations between repetitions in an audio stream, it poses a great challenge to information retrieval. The classical approach consists of computing the self-similarity distance based on (Foote 1997) between all existing audio analysis frames of a stream, and undertaking machine learning methods most often governed by musicological heuristics to obtain such structures. Figure 2.6b shows such a self-similarity matrix over a performance of Beethoven's Piano Sonata N.1 by the renowned Pianist Friedrich Gulda in 1958[1]. The intensity of each value on the image matrix shows the degree of similarity between the two audio instants on the two axis (which are the same). Besides the main diagonal which reveals the self-similarity, parallel darker lines illustrate repeating structures over the audio stream. Having this representation, much further processing is needed to deduce structures and recurrences and existing approaches differ at this stage: dynamic programming and pattern matching (Chai 2005), multiple-pass Hidden Markov Models with musicological priors (Peeters 2004), K-means clustering (Logan et al. 2000), or direct image analysis of this data (Foote and Cooper 2003) to cite a few. Note that besides cascaded methods, these approaches are trivially non-realtime and non-causal, often quite costly and prone to errors due to hand-engineered priors.

By modeling audio streams as quasi-stationary chunks where each segment is a model over a flat Riemannian manifold as shown in the previous section, one can imagine employing the sequence of models as a time-series for representing the audio stream instead of samples. Using this representation makes data and its processing sparser and brings in the idea of profiting from the geometrical entities to define equivalence between classes and in return be able to deduce recurrences online.

Following this scheme, we have shown in (Cont, Dubnov, and Assayag 2011) that high-level extraction of symbolic data and their representation within incremental signal, can become trivial in realtime if the audio stream is directly represented on information manifolds constructing a *similarity metric space*. The idea is to do proper mathematical manipulations to construct a *metric space* that exploits an information theoretic definition of *similarity* on a statistical representation of audio streams. Moreover, the temporal order of these statistical *points* on the information manifold is taken into account to construct sparse state-space structures where each state corresponds to *metric balls* on a Riemannian manifold that correspond to quasi-stationary and continuous audio chunks. Once this *similarity metric space* is given with their geometrical constructs (such as the notion of equivalence classes in analogy to the symbolic domain), the hard task of detecting repeating structures is a simple adaptation

[1] This excerpt is chosen on purpose because of its low quality recording and extreme temporal variations between repetitions by the performer.

of any text/symbolic algorithm to this new metric space. Figure 2.6a shows the result obtained incrementally on the same audio as in figure 2.6b. Intensities on this image matrix correspond to arrows of the learned state-space structure. Therefore, we can address this rather non-trivial problem in one shot and by issuing the fundamental question of information representation. Note also that figure 2.6b corresponds to a 9600×9600 matrix computed offline, whereas that of figure 2.6a is a 440×440 sparse representation computed incrementally as data arrives to the system, for the specific chosen piece of music.



(a) Information Geometric Similarity Matrix based on (Cont, Dubnov, and Assayag 2011)

(b) Traditional Similarity Matrix based on (Foote 1997)

Figure 2.6: Information Geometric Similarity Matrix on Beethoven's first Piano sonata (Recording by Friedrich Gulda 1958) using methods of information geometry (left and incremental) and traditional self-similarity computing (right and non-causal)

The elegance of information geometric approaches is not limited to simplifying complex problems such as the example above. Information geometry is at the intersection of several scientific disciplines: information theory, statistical learning, and differential geometry. The beauty of information geometry also lies in the fact that many commonly used but disjoint approaches to pattern recognition and optimization problems converge towards one single and intuitive concept on information manifolds. For example, computing the centroid of entities on such manifolds might coincide with the definition of maximum likelihood and maximum entropy estimations over the dataset. More interesting, convex optimization problems on the natural parameter spaces will become linear for expectation parameters on the dual space of such structures based on Legendre transformations. In certain conditions, mixture densities would become also a dual representation of their parent distribution manifold (Amari et al. 2000). Note that these are all engineering tools presented in various engineering applications and united within one single and elegant theoretical framework.

Despite these encouraging results, much fundamental work remains to be done for further formalization and construction of information manifolds over temporal streams. For example, the information geometric construction for fig-

ure 2.6a leads to metric balls with different radii and represent the entire audio stream. Are these metric spaces compact? Could these metric balls be considered as ϵ-nets covering the entire compact stream? Can we achieve in the same manner *length spaces* between information entities (Burago et al. 2001)? Once audio entities are represented on these manifolds, how would smooth Riemannian length structures and their corresponding geometrical transformations actually affect the sound once synthesized? What is the effect of parallel transport on an audio entity once resynthesized? Could this be a new paradigm for temporal audio analysis and resynthesis? Can these models extend to non-parametric distributions? The answer to such question lies within coarse and long-term research where, as is the case up to now, focus is on adopting new representations of incremental audio for complex engineering problems with intuitive and simple solutions.

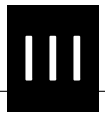
This aspect of *machine listening* research is the most theoretical and will occupy our research efforts in the years to come. We collaborate with colleagues working in other application domains and employing similar techniques. The author acted as Founder for the *Brillouin Seminar Series* at Ircam with THALES and LIX as partners and bringing together more than 60 researchers and organizing seminars[2]. In 2011, we organized three scientific events for dissemination of the group's synergy: The *Matrix Information Geometry Workshop*[3] sponsored by CEFIPRA, a special session during the *SMAI 2011* symposium[4] addressing issues in the field of Applied Mathematics, and a special session on signal processing and information geometry during the *GRETSI 2011*[5] conference. It has come to our attention, within our Brillouin seminars at Ircam, that similar approaches can lead to interesting applications in audio. Among them are the use of Wasserstein spaces as proposed by Xia et al. for dynamic image texture analysis (Xia et al. 2012), and group invariant scattering representations in (Mallat 2011). These approaches, along with information geometry, will be studied and situated for time-series analysis of incremental audio.

[2] See http://repmus.ircam.fr/brillouin/ for upcoming seminars and videos of past seminars.

[3] http://www.informationgeometry.org/MIG/

[4] http://smai.emath.fr/smai2011/

[5] http://www.gretsi2011.org/



REACTIVE SYNCHRONOUS PROGRAMMING FOR COMPUTER MUSIC

This chapter summarizes our efforts since 2008 in formalizing and developing a Reactive Synchronous Language (Halbwachs 1993) for Realtime Interactive Music repertoire of mixed electronic music. The developed language in its current form is integrated in Antescofo, in charge of executing computer actions as a reaction to the realtime machine listening described in section 2.2. The listening module in Antescofo is restrained today to a score following module (or live automatic alignment) of live musicians with a score, but the language described here has its merit in wider application contexts. In this chapter we discuss motivations for the development of the Antescofo reactive synchronous language, its current form and usage in the mixed electronic music repertoire along some prospective research. The extension of the language to the wider context of *Cyber-Physical Systems* is postponed to chapter 4.

The application context of this chapter thus targets the existing repertoire and new pieces created for *Mixed Electronic Music*, defined as the live association of human musicians with computers. We first review existing practices for authoring, implementation, software architectures and performances of this repertoire. The following review is far from being exhaustive but is representative of common world-wide practices since the 1990s.

3.1 Computing and Programming of Mixed Music

Mixed Electronic Music refers to the live association of human musicians with computers (and possibly other external mediums). The difficulties and challenges posed by this mixture are on two fronts: (1) the confrontation of the two heterogeneous (human and artificial) mediums during authorship, and (2) temporal juxtaposition of processes during realtime evaluation or performance. The absence or difficulties in explicitly exploiting interactions between the two mediums have challenged composers, musicians, as well as computer scientists for decades. To this end, various musical practices and solutions have evolved since the 1950s (See Cont 2012, for a review) and various computational paradigms

and software solutions (Assayag and Gerzso 2009).

Among common musical practices in the domain, the dominating practice today belongs to the so called *school of realtime music* initiated at Ircam originally by composers such as Philippe Manoury, Pierre Boulez and Brice Pauset (Szendy 1998) along with attempts to theorize the advent of realtime processing in this new practice of music composition and performance notably by composer Philippe Manoury in (Manoury 1990). This formalization, in collaboration with Miller S. Puckette, led to new software and programming paradigms for such practices (Miller Puckette 1988; Miller Puckette 1991), paving the way for two dominating programing environments Max (Cycling74 2012) and its open-source counterpart PureData (Miller Puckette 1997). We will discuss technical insights on this in section 3.1.2 and refer the curious reader for details to (Miller Puckette 2002).

On a less technical basis, responses to the challenges posed by Mixed Electronic Music has led to a divide in general practices of computer music between *Performative* and *Compositional* aspects of computer music (Miller Puckette 2004). This situation has led to specialized softwares in respective domains whereas the very practice of computer music demands for both. We will come back to this crucial question in section 4.2.1.

In order to better understand and analyze the challenges of mixed electronic music, it is important to study the workflow of the creation of such pieces. In light of the introduction above, we look at this workflow in three stages: *Authoring* consisting of the authorship for all mediums required by the composer to create such pieces involving language formalization and computer implementations, *Software Architecture* in common use within the repertoire, and *Performance* phase or the actual practice of a mixed piece.

3.1.1 Authoring

Authoring of mixed electronic pieces, loosely speaking, consists of the *compositional* phase where both mediums of computer and instrumental actions are put together in some forms of writing to prepare performances, implementations, and transmission of the piece. For the sake of this study, the process of authorship begins when the creator is in the possession of the necessary vocabularies constituted by the materials in hand and consist of assembling and choreographing them altogether to create the global work. The process of authorship is a reflective process (Donin 2012) but this fact does not alter our study. This vocabulary might vary from composer to composer and from piece to piece and consists of instrumental techniques as well as special computer programs and combinations thereof for a specific piece. Despite this variety of basic vocabulary, we argue that the *language structure* that brings these elements together have much in common across personalities and pieces.

More specifically, a mixed electronic music piece consists of computer programs that are put next to and parallel to instrumental events. The computer programs implement various modules with possibly different models of computation that handle a variety of possibilities in computer music (see section 4.3.1) such as: sound spatialization, live audio transformations from the musician,

sound synthesis, sound analysis, and gesture processing just to name a few. Some of these programs employ realtime computing in interaction with the live musician (e.g. audio transformations), whereas others are essentially offline oriented algorithms (e.g. sound synthesis) but must be triggered online in a live performance situation. Besides computer programs and modules that materialize electronic actions, a mixed piece is often equipped with a *score* as a guideline for the temporal realization and juxtaposition of different parts. Such scores are in direct analogies to classical music scores but can take various forms and directions depending on the style of the composition.

Figure 3.1 shows a hand-written score manuscript of `Tensio` for string quartet and live electronics by Philippe Manoury (2010) as a result of authorship. Similar to a classical music score, each staff (or line) represent different voices and the horizontal line represent (musical) time. The score for (human) instruments (lower four staves) is written synchronous to that of computer actions (top staves) despite their different nature. Each computer staff is labeled with its appropriate program module specifying the nature and destination of the task in the computer program, as well as the temporal progression of their parameters along instrumental events. Note here that the temporal nature of computer actions are heterogeneous, employing continuous lines and discrete events. Obviously, the score of figure 3.1 implies synchrony between all actions and despite the variability of the interpretation.

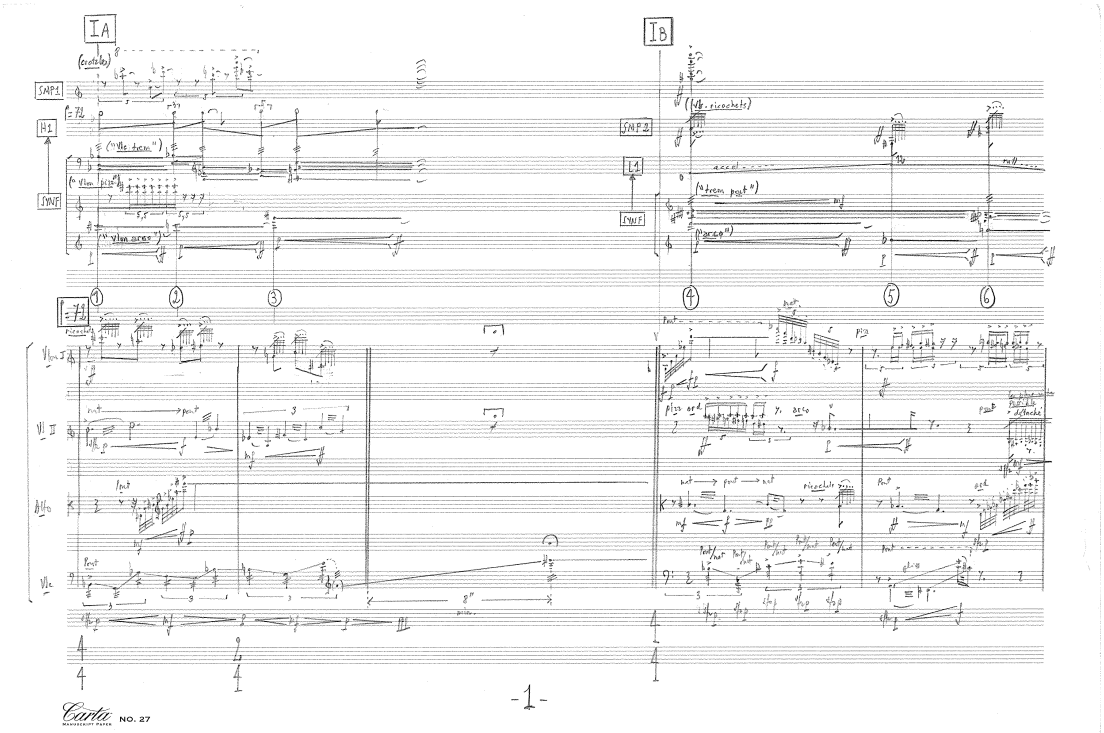


Figure 3.1: First page of `Tensio` for string quartet and live electronics by Philippe Manoury (2010). Lower group of staves are intended for a string quartet whereas the top staves indicate live electronic actions.

Figure 3.2 shows another example from the published score of *Anthèmes II* composed by Pierre Boulez for violin and live electronics (1997). This music score shows the instrumental (violin) section in parallel to an approximative notation for the realtime electronics accompanying the system. Each system corresponds to a specific electronic process, whether realtime or samplers, by themselves accompanied by spatialization parameters. The sequencing of electronics in this score are either notated as relative to the performance tempo or fixed absolute values if necessary. The circled numbers in the score correspond to *synchronization points* between the instrumental and the electronics scores.

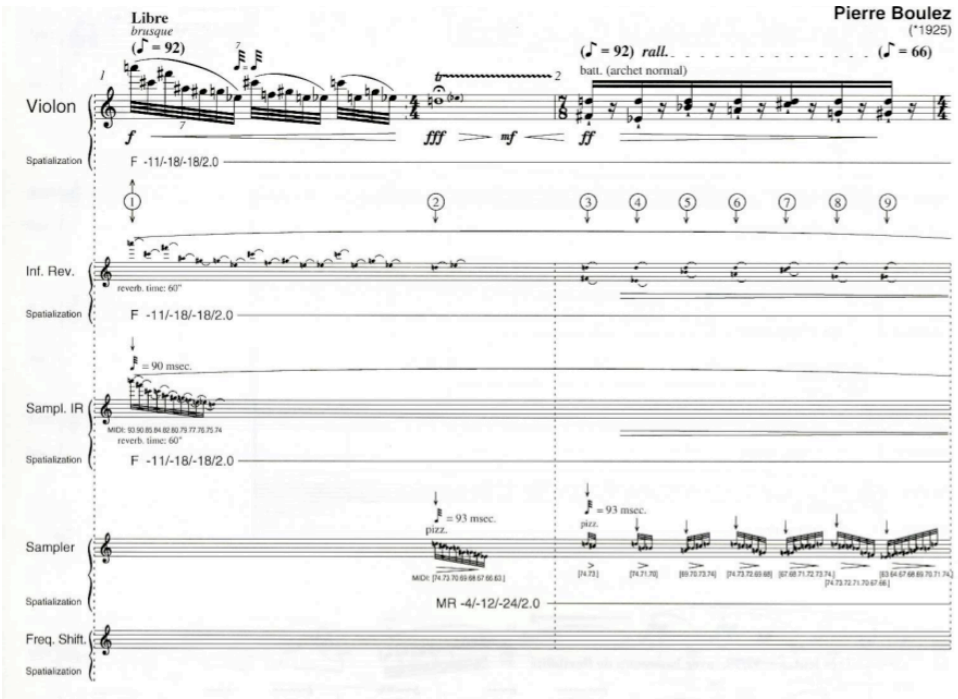


Figure 3.2: First page of `Anthèmes II` for violin and live electronics by P. Boulez (1997). Top staff shows the violin score, lower staves are electronic actions.

To a typical music reader, the scores of figure 3.1 and 3.2 imply the general organization of the piece (similar to an orchestral score) as well as expected actions from the musical instruments. However, the amount of information exposed for computer actions might still need clarification. Despite the apparent superior amount of information in the instrumental part, it is important to note that much valuable information responsible for final evaluation, is still missing in the instrumental part such as event articulations, specific nuance progressions and tempo variations. Some of these parameters are described relatively (such as violin nuances in figure 3.2) or left to the human interpreter. In other words, much of the final outcome parameters are either hidden or loosely represented by they are all *virtually* present. This virtuality in music authorship is present in almost all styles of written music. It is precisely this virtuality that can tie the two worlds of human performers and computer actions together as proposed

by Manoury and his notion of *Virtual Scores* (Manoury 1990)[1].

A virtual score is a musical organization in which we know the nature of the parameters that will be processed but not their exact outcome at runtime since they're expressed as a function of the live performance. A virtual score hence consists of electronic programs with fixed or relative values/outcomes to an outside environment. A realtime electronic process is therefore one that exists in a music score, next to the instrumental transcription, and whose outcome is evaluated during live performance and as a function of the instrumental part's interpretation with all its diversity and richness.

The idea of virtual score is thus to bring in both the performative and compositional aspects of computer music within one compositional framework. In the case of scores in figure 3.1 and 3.2, the electronic actions are thus virtual since their final outcome depends entirely on the realtime context despite bring entirely authored; analogous to instrumental authorship. Some parameters are fixed (e.g. sampler pitches in fig. 3.2 and synthesizer ranges in fig 3.1) whereas others will be a function of runtime evaluation (e.g. final spatialization outcome and final evaluation of synchronous rhythms).

The framework of *virtual scores* is at the core of most interactive programming environments in computer music today. Despite its similarity to a traditional framework of composition, it does not limit its practice to traditional norms of music composition and on the contrary it has integrated non-traditional practices of computer music such as interactive composition (Chadabe 1984), hyperinstrument composition (Machover et al. 1989), composed improvisations (Chabot et al. 1986) and more, as employed in Manoury's and Boulez' early realtime pieces among others (Miller Puckette and Lippe 1992; Boulez et al. 1988).

The final outcome of scores in figure 3.1 and 3.2 depend on the instrumental interpretation for one end, and the realtime evaluation of computer programs in reaction and interaction to the musical environment on the other end. We now attempt to study the architecture of such programs.

3.1.2 Software Architecture

The authorship of a mixed electronic piece is closely tied to its programming. The score, as seen previously, calls for synchronous calls to specific modules at each instant associated to events coming from the environment (human performers). Each module takes the form of a program (or a set of inter-connected programs), and the main program is a collection of these modules tied together using a representation of this score and in interaction with the environment.

[1] Curiously, the electronic musician Brian Eno refers to the same process expressed in an interview in *WIRED*: "I realized that "interactive" anything is the wrong word. Interactive makes you imagine people sitting with their hands on controls, some kind of gamelike thing. The right word is "unfinished." Think of cultural products, or art works, or the people who use them even, as being unfinished. Permanently unfinished. [...] the "nature" of something is not by any means singular, and depends on where and when you find it, and what you want it for. The functional identity of things is a product of our interaction with them. And our own identities are products of our interaction with everything else." (Kelly et al. 1995)

Among different platforms for this kind of programming, Max and PureData are the most commonly used. They are both realtime graphical programming languages based on a combination of hybrid event and signal processing scheduling (Miller Puckette 1991). Programing in Max and PureData amounts to connecting boxes together in a top-down manner. Each box can be another program (or *patch*) with proper communication, or a C/C++ object compiled using the proper API.

Figure 3.3 shows the main patcher for Manoury's `Pluton` for Piano and live electronics (1988) in PureData[2]. This main window is basically a collection of all DSP, Control and interaction modules required for the piece to run. Modules communicate together by the means of send and receive signals that can be signal or control.

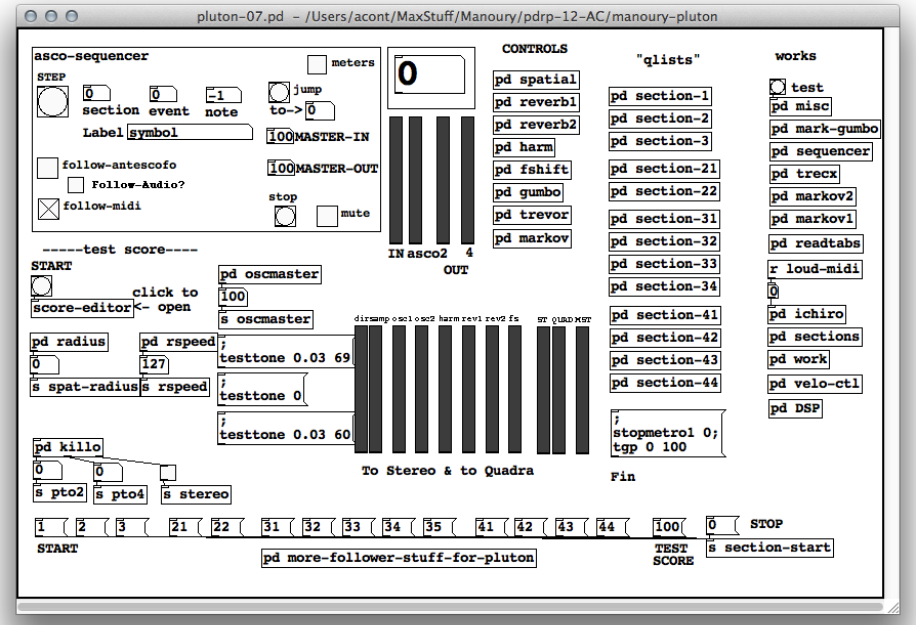


Figure 3.3: Main patch of `Pluton` by Ph. Manoury for Piano and Live Electronics (1988), in PureData.

Figure 3.4 shows the DSP patcher of the main patch in figure 3.3. This patcher contains all the DSP modules for transforming, treating, analyzing and producing sounds in realtime. The bottom-right window in figure 3.4 shows the implementation of the `[pd freq-shift]` box in the DSP patcher which consist of a classical frequency shifting of the realtime audio using Hilbert Transform and some receiving parameter. Note that the main DSP patcher allows passing and chaining results altogether if asked by the score.

What ties the DSP and commands altogether during performance (or time) is a formal representation of the scores similar to those shown previously. In this

[2]This piece is supposedly the first Max program!

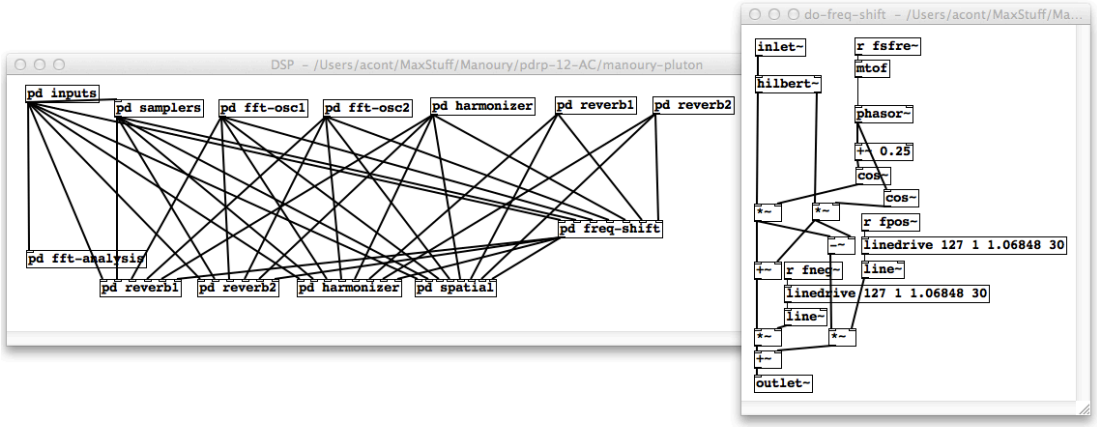


Figure 3.4: Portion of DSP patch in `Pluton` by Philippe Manoury for Piano and Live Electronics (1988), in PureData.

(historical) piece, the evolution of control parameters are represented as simple message-passing hooked to pre-defined *synchronization points* in a score (e.g. circled numbers in figures 3.1 and 3.2) through a dynamic select mechanism. This is illustrated in figure 3.5 for events ① and ② of section 2.

The sequence representation using messages and absolute delays as in figure 3.5 are aimed at representing or approximating a score such as ones in figures 3.1 and 3.2. An important downfall of this representation is a severe semantic decline of the rich representations in the general score with regards to synchronization and timing. In the above example a delay object (accepting time in milliseconds) is employed for implementing sequencing inside an event where as most timings in such pieces are relative to the pace (or tempo) of the musician. Synchronization between the two mediums will also be degraded to the indicated synchronization points whereas most scores are composed having a continuity in mind (best seen in the score of fig 3.1).

The patches in figure 3.3 to 3.5 show the general architecture used for the implementation of the performance program of `Pluton` (1988) according to its score and along with a human Pianist.

Surprisingly, modern patches or programs for recent pieces in the repertoire follow the same architectural principles despite enjoying better GUIs, more high-level objects (e.g. for signal dispatch instead of the "spaghetti patch" of fig. 3.4) and possibly more advanced signal processing and control objects. To make things simple, while computer music research has greatly contributed to the enhancement of new vocabularies and their integration in digital art environments, little progress has been made addressing more comprehensive ways for programing and executing programs such as the one showcased above and despite significant advances in the general literature of realtime systems.

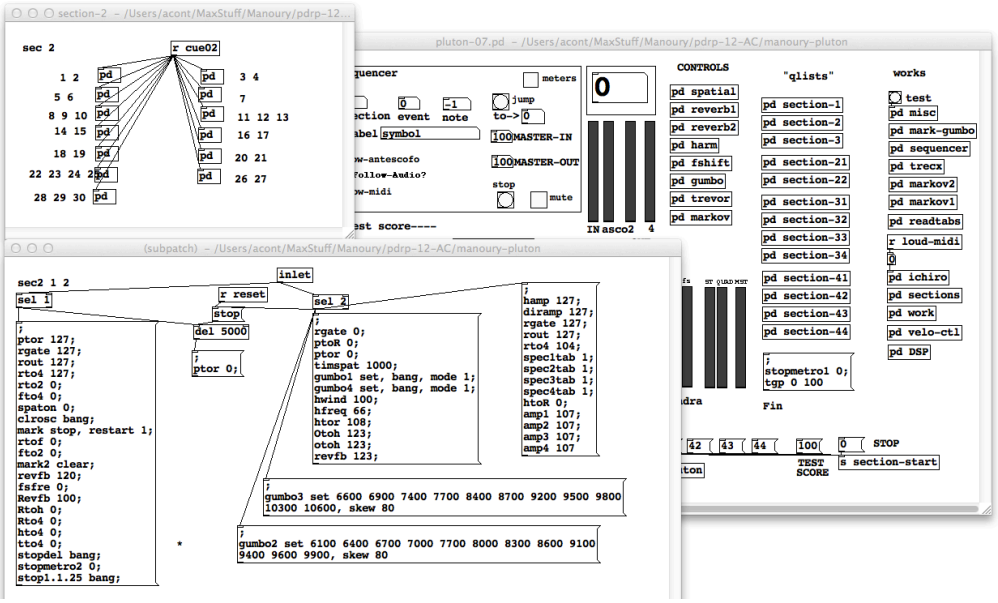


Figure 3.5: A control patch of `Pluton` by Philippe Manoury for Piano and Live Electronics (1988), in PureData.

3.1.3 Performance

The programs of mixed electronic music pieces shown above are also in charge of realtime performance and interaction with musicians during concerts and rehearsals. On the computational level, a system of this kind requires timely triggering of actions and events synchronous to live musicians and following a score, and timely delivery of their inputs/output according to the nature of the computational model for each action. While interacting with the outside environment, the behavior of the system is expected to be *deterministic* based on the specifications, ordering and synchronicity provided in the score. Moreover, it is expected that the performance system be *fault-tolerant* with regards to noisy and faulty environment during live performances. Such a system can thus be considered *critical* similar to critical realtime systems in avionics, despite the fact that system crash down in a concert usually does not imply human casualties.

Performance of pieces such as the ones presented above can be either automatic through dynamic machine listening or interactive analysis of the environment (such as in score or gesture following), controlled by a human operator, or semi-automatic using a combination of both. In all combinations, expected temporal behavior of a mixed score remains the same.

Despite their common use, Max and PureData can become non-deterministic on the computational level. The scheduling mechanism in both is similar to periodic Round Robin mechanism for time-sharing between DSP and control computation where signal processing is preemptive with regards to control (for on-time delivery of audio buffers). Despite apparent concurrency in a Max program, the engine does not support concurrency. All patchers lead to a chain of

calls (either DSP or Control) where ordering is top-down and right-left[3], these queues are flattened for the time-shared scheduler. DSP computation cycle is based on fixed buffers of minimal 64 samples. The scheduler is *not* aware of any explicit notion of duration for processes nor timing requirement. The main ambition is to get DSP buffer outputs delivered on-time. It is therefore not surprising to expect jitter at instances of high-computation and non-deterministic delays at points due to DSP preemption of control since the scheduler is not dynamic.

Max and PureData provide neither features for *critical safetiness* nor *fault tolerance*. This is mostly due to the lack of explicit models of time both semantically and syntactically in both environments. To demonstrate this consider again the score in figure 3.2 as an example: The first issue is clear by differentiating electronic events tagged by ① and ③. Event ① does not necessarily depend on the recognition of the first note in bar 1 (an F) while event ③ can be considered as having a local scope. If the musician or the recognition system misses the first high F in the violin part, it is evident that ① should not be dismissed. This is however not true for ③. If the chord corresponding to ③ is missed, that event can be consequently dismissed in order to maintain a musical coherence in the output. On the other hand, if some consequent action in ③ is a prior to (for example) ④, that portion of the program cannot be missed whereas local music events can.

3.2 Antescofo: Performance-Synchronous Language for Computer Music

Antescofo is a software and a dedicated programming language developed initially by this author for synchronization and control of interactive parameters in computer music (Cont 2008a). Antescofo is currently developed as an external object for graphical realtime programming environments Max and PureData for best integration in existing workflows. It aims at making the practices of authoring, programing and performance as smooth and logical as possible within the workflow presented in the previous section. The language interpreter is coupled to an artificial listening machine, capable of realtime decoding of musicians' position and musical interpretation parameters (such as speed and dynamics). Antescofo is thus a coupling of both a realtime score-following system (described in Section 2.2) and a synchronous programming language for computer music presented here.

The philosophy behind the development and coupling of Antescofo since its inception consists of:

- *Bridging the gap between performative and compositional aspects of computer music,* by integrating the dynamics of the environment (musicians on stage) into the core design of the system and bypass the existing separation in modeling.

[3]The horizontal order of evaluation in Max and PureData are subtly different; we leave that out for simplicity here.

- *Authoring of Time and Interaction for Computer Music,* by providing explicit languages for expressing complex temporal and vertical structures as seen in section 3.1.1.

- *Bringing the richness of time models in music to programming,* where temporal relationships and inter-relationships are many whereas their transitions to computer programs as seen in section 3.1.2 have been severely underestimated.

- *Creating a dedicated literature for Realtime Computing in the Arts,* following the long tradition of realtime embedded systems (E. A. Lee and Seshia 2011) and their long studied design consequences such as critical safety (Gérard Berry 2003) and fault-tolerance (Kopetz and Bauer 2003) by adapting them to needs and formalizations of current and future computer art practices.

Since its inception in 2008, Antescofo has enjoyed numerous collaborations with various artists and has led to the creation of more than 40 original mixed electronic pieces at Ircam alone, with performances with world-renown artists and ensembles such as New York Philharmonic, Berlin Philharmonic, Los Angeles Philharmonic, Radio France Orchestra and more. Antescofo was the laureate of the "La Recherche" Jury prize in 2011. Its development is incremental and adapted to challenges posed by each project, where the artistic imagination meets scientific creativity. This process is the main drive in the development of this project, creating new horizons for art/science research.

In the following sections, we attempt to explain various aspects of the system and the language borrowed from (Cont 2008a; Cont 2011; Echeveste et al. 2011; Cont 2012; Cont, Echeveste, et al. 2012; Echeveste et al. 2012).

3.2.1 Informal Description of the Language

In Antescofo a program is text describing the dynamics of both the *environment* (acoustic environment outside the system) and *computer actions*. The idea is to represent a mixed score as discussed in section 3.1.1 in its entirety. These programs are thus *synchronous reactive* with respect to both represented dynamics and the goal of the system is to respect specifications during realtime performance with the ultimate goal of *determinacy* (in the computer science sense), correct ordering at runtime, and (musical) critical safetiness. Such paradigms have been widely studied in the computer science literature for *realtime synchronous languages* (Halbwachs 1993) and widely applied in the industry for realtime critical systems. We informally describe both aspects of the Antescofo language. For a formal definition refer to (Echeveste et al. 2012).

Environmental Dynamics

In a mixed music setup, the dynamics of the environment can be described by the expected musical elements that will be fired by the acoustic environment into the system. This includes musical events described in the instrumental

section of a score. In the current version of Antescofo, the system accepts western musical notation and its variants and extensions . For the environmental dynamics, current Antescofo syntax represent the content (usually pitches) as well as durations. This will lead to the construction of an automata as described in section 2.2. For example, the program corresponding to the state-space of figure 2.2 is represented here in figure 3.6. Other supported event types include

```
; This is a comment
BPM  60
NOTE D5    0.0           ; grace note
NOTE C5    1.0    note1  ; regular note
NOTE D5    0.0
NOTE E5    1.0    note2
NOTE 0     0.5           ; silence
```

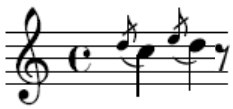


Figure 3.6: Example of environmental dynamics automata in Antescofo.

TRILLs, CHORDs, and continuous events notated as MULTI, plus tempo changes with or without modulations (See Cont 2008a, for details).

Action Dynamics

In addition to the environmental dynamics, an Antescofo program specifies the computer actions at each instant and associated to (known) environmental events. Actions in Antescofo are simple messages sent to defined destinations and at the right time in realtime. With this respect, Antescofo acts as a *message-passing coordination language*. An action is a line anywhere in the program with the following syntax:

```
<delay> <receive-symbol> +<message> !<attribute> !<name>
```

where <delay> is optional and can be either a float relative to running tempo, a rational number (relative to tempo, or absolute time (float followed by 'ms' or 's'), or a an expression over a variable for runtime evaluation; <receive-symbol> is the symbol indicating the destination *receive* in Max or PureData; +<message> is a list of values accepted in Max or PureData[4] plus calls to *functions* that are evaluated at runtime; <attribute> for now indicates whether the action is *local* or *global* (specification for fault-tolerance); and finally an optional <name> for the action useful for controlling periodic or looped content.

An action as described above is the primary element of the language. Consequent lines of actions can thus describe a simple sequence whose timings can be relative to a tempo, in absolute time or a mixture of both.

On top of primitive actions, the Antescofo language compositionally provides the following constructions using primitive actions:

Parallel Group (GFWD) Actions can be *grouped* to constructively represent a single grouped action in a sequence. This semantic provides polyphony and parallel actions during authorship as well as independent but relative timing between groups of electronic phrases. A group can be given a name.

[4]Int, Float, Symbol and List. PureData represents Int and Float as Float.

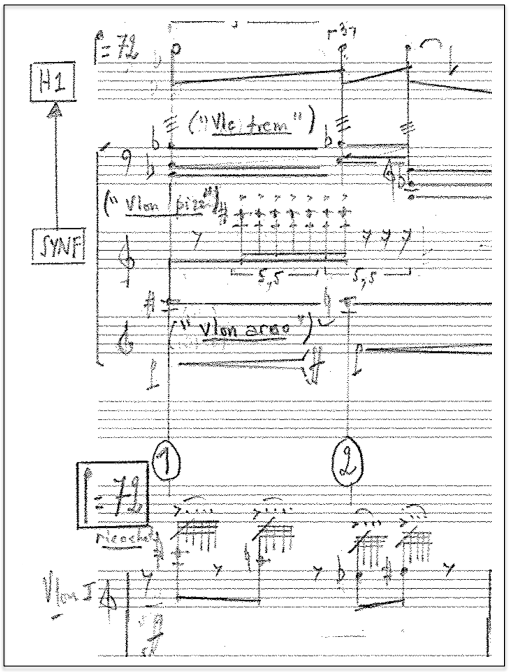
Periodic Group (LFWD) A *group* equipped with a global (absolute or relative) period, useful for constructing *loops*, that run forever after instantiation unless KILLed by its name.

Continuous Group (CFWD) A *group* equipped with a global (absolute or relative) step-time, that considers its internal atomic actions as break-points and fires actions using interpolation and continuously inside the process using the given time-step. This construction is useful for representing continuous voices.

All constructions can be nested hierarchically indicating timing paths. Macros are available and expanded at score load and useful for programming motivic patterns.

Composed groups above can optionally accept attributes for synchronization strategies and error handling. These attributes indicate whether long phrases are tightly or loosely synchronous to environmental events once launched, and are critical for understanding the behavior of actions or groups in case of an error (from the environment or system). Additionally composed groups can accept their own tempo (clock) as expressions bound to the environment or variables (See Echeveste 2011, for details).

Example




```
BPM     72
TRILL   ( 8500 ) 0.5            IA...EVT-1
GFWD            arco
{
        syn_voices 1    pgm 40  bnk 7   vol 122
        syn_voices 1    56      120     @b(2)
        syn_voices 1    50      120     @b(2)
        syn_voices 1    46      120     @b(2)
2.0     syn_voices 1    53      120     @b(0.36)
        syn_voices 1    52      120     @b(0.36)
        syn_voices 1    58      120     @b(0.36)
0.36    syn_voices 1    69      120     @b(1.5)
        syn_voices 1    63      120     @b(1.5)
        syn_voices 1    59      120     @b(1.5)
1.5     syn_voices 1    66      120     @b(0.8)
        syn_voices 1    65      120     @b(0.8)
        syn_voices 1    62      120     @b(0.8)
}
CFWD        h1_trans    @grain 30ms
{
        0       0
        2.0     300
        0       0
        0.46    800
        0       0
        1.5     -500
}
GFWD    0.5     Pizzicati
{
        Bsyn_voices     4       vol 127
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
1/5.5   Bsyn_voices 4   85      127     100
}
NOTE    0       0.5
TRILL   (8100)  0.5
NOTE    0       0.5
TRILL   ( 7500) 0.5                     IA...EVT-2
```

Figure 3.7: Antescofo score example from *Tensio* for string quartet and live electronics by Philippe Manoury (2010).

Figure 3.7 shows an Antescofo program of an excerpt of *Tensio* in figure 3.1. The left column shows the original manuscript as composed by the artist and the right column the Antescofo counterpart. In this example, the lower graphical staff corresponds to the first Violin events and the higher staffs to the description of electronic actions. The violin events are notated as `TRILL` events (for violin tremolos) and silences (`NOTE 0`) with duration relative to an initial tempo of 72 beats-per-minute. Electronic actions in this example are hooked to the first instrumental event and described in three parallel groups entitled `arco`, `h1-trans` and `Pizzicati` in analogy to the polyphonic authorship in the left column. Each line in a group describes a single action as simple message-passing to a receive signal in the general environment with optional delays. In this example, each group describes a sequence of ordered actions relative to the live tempo of musicians. The two groups `arco` and `Pizzicati` contain *discrete* and atomic sequences (defined as `GFWD`) whereas the `h1-trans` (defined as `CFWD`) contains a *continuous* sequence. This continuous process interpolates between break-points (first element being time and followed by values) using a time-grain of 30 milliseconds. The continuous process `h1-trans` correspond to the top staff in the left column. The time nature of each sequence (discrete, continuous, relative or absolute) is left to the programmer/composer discretion and depends on the nature of the processing at the receiving end.

3.2.2 Runtime Semantics

Antescofo provides an abstract programmer's model for an artificial musician in an ensemble with musical realtime constraints. Its runtime system is composed of two virtual machines. The first one is the *listening machine*, the score follower of section 2.2, that handles the interaction with the environment (reactive). It interprets the *environmental dynamics* written in the program and supervises the execution of action tasks in response to the physical events and passes on necessary parameters such as the environment's tempi. The second one is the *Scheduling Machine* and handles the interaction with the running platform (Max and PureData). The Scheduling Machine is *anticipatory* and *proactive* and interprets the action dynamics in the program which specifies the temporal order of actions. Figure 3.8 shows how the *Listening Machine* and *Scheduling Machine* interact with the physical environment, software task and hosting platforms. Hollow arrows indicate offline whereas filled arrows indicate runtime processing.

Antescofo programs are not compiled but interpreted at runtime. The semantics of the system thus can only be studied as a dynamic system. An Antescofo program automatically creates two dynamic automates required for the two virtual machines. The listening machine is based on the system described in Section 2.2 and communicates the detected events $e(t)$ and their tempo or pace of arrival $\dot{e}(t)$ to the scheduling machine. The detected tempo $\dot{e}(t)$ can be considered as the inferred *clock* of the environment which is used by default as the clock for all underlying processes unless stated in the program. Actions in Antescofo are hierarchically parsed into entities called *Trigger Timers (TT)* for each level and group type as FIFO stacks of actions. An action can be an elementary action or refer to another Trigger Timer. As an example, fig-

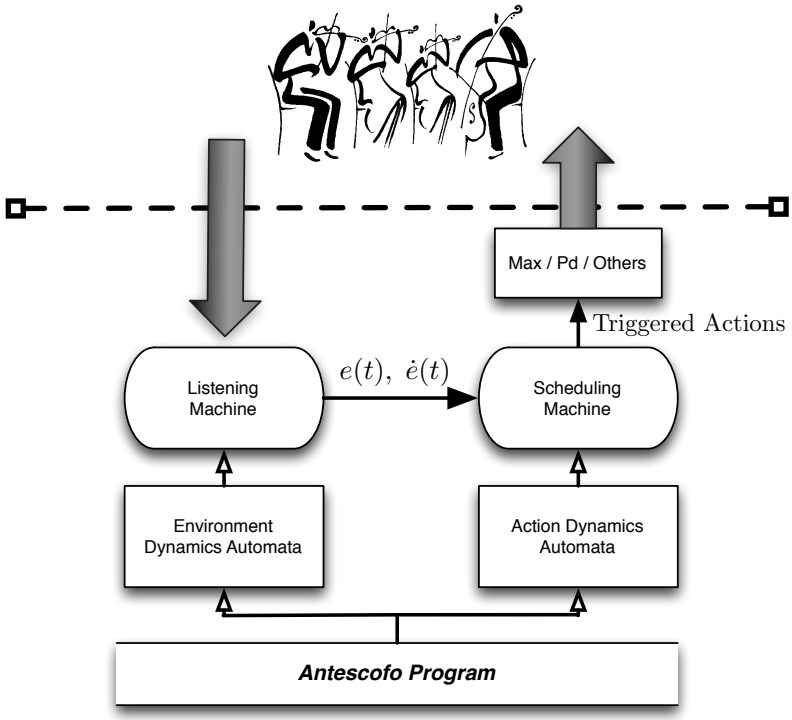


Figure 3.8: Components of Antescofo runtime.

ure 3.9 shows a simple Antescofo program and its equivalent TTs. For the sake of simplicity, actions are shown as pairs of delays d_{ik} and abstract actions a_{ik}. High-level Trigger Timers such as TT_{i0} are hooked initially to an associated event (e_i here) whereas low-level TTs such as TT_{i1} here are relative to their high-level Timers. Nested TTs indicate that at a deterministic moment in time, a (low-level) TT will be launched in parallel to existing ones. Group attributes (local, global, tight, loose) for lower-level TTs are inherited from their parents unless stated.

Each Trigger Timer type has its own associated runtime semantics in the scheduling machine. We showcase the operational semantics of two major types using Timed Automata (Alur et al. 1994) and following notations in (E. A. Lee and Seshia 2011).

Figure 3.10 shows the operational semantics of *High-Level Timer-Triggers* in the scheduling machine (for example TT_{i0} in figure 3.9). The initial condition for entering this automata is either at the detection of its corresponding event e_i or if the pointer to this event has passed (missed event, detection error). The automata is equipped with a clock $\dot{s}(t)$ which for high-level TTs is always equal to $\dot{e}(t)$. The operator *Pop* simply pops out the last element in the TT FIFO. Obviously this process will eventually empty-out the FIFO. An action a_{ik} is launched if its corresponding delay d_{ik} has been reached within the scope of e_i (the top self loop). Since the scope of high-level TTs are bound to e_i, in case of its absence, they pass to the *Critical Mode* where global actions are undertaken immediately, in order, and before any consequence $TT_{(j>i)0}$ actions.

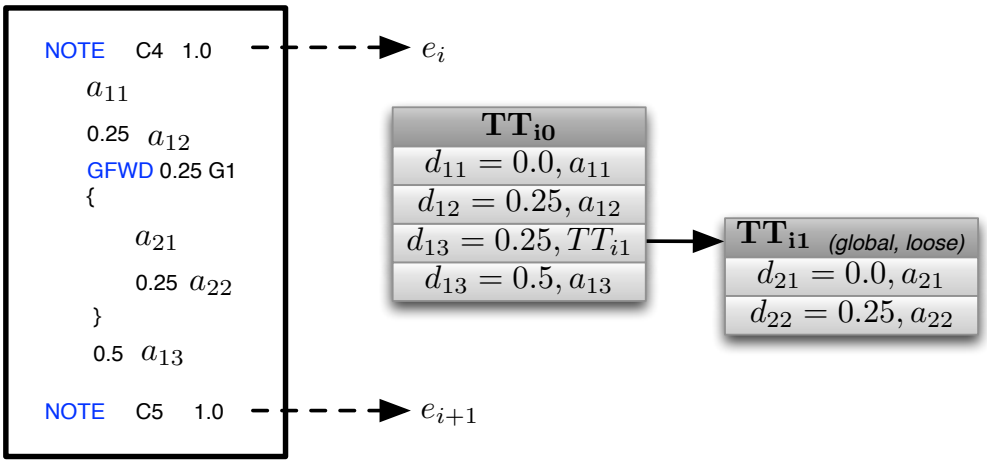


Figure 3.9: Example of *Trigger Timer (TT)* construction in Antescofo.

The *Terminal State* is reached when the running FIFO is emptied or a *KILL* is requested (either directly or from a parent).

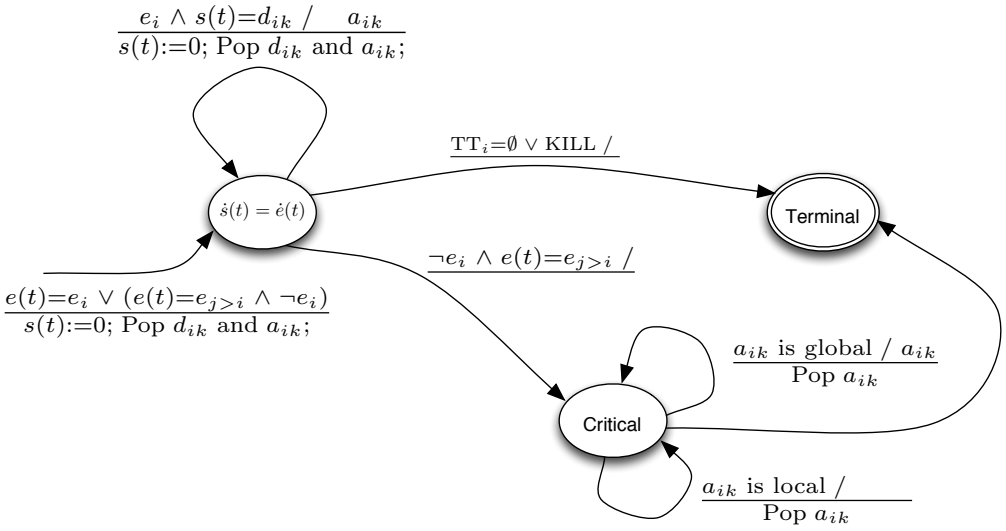


Figure 3.10: The high-level Timer-Trigger runtime semantics.

Note here that the passage of time indicated by $\dot{s}(t) = \dot{e}(t)$ is not that obvious since $\dot{e}(t)$ changes neither synchronously to actions, nor to event steps from the environment. This implies the implementation of an interrupt rescheduler on top of these models that at each change of dynamics reevaluates active parameters. To this fact we should also add the fact that delays can become runtime expressions following similar mechanism during scheduling.

Figure 3.11 shows the operational semantics for a `tight GFWD` TT. A `loose GFWD` has a runtime semantics similar to figure 3.10 since it only synchronizes with tempo once launched, which we skip. A `tight` group has the specificity that on top of a dynamic clock, it also synchronizes to positions (or *dates*) of events during performance. Scheduling is similar to figure 3.10 if the action scope

does not go beyond the future $e(t)$. Therefore, an auxiliary variable U traces the instantaneous event index and the behavior of this automata is proactive. If the date of the current action, according to the score, is superior or equal to its future event, the scheduling machine neglects the clock and waits until it becomes useful (after e_{U+1}) by changing the consequent delay. The *Critical Mode* here is similar to fig 3.10 except that it only handles actions up to the next future event. In other words, the critical mode for `tight` attempts to *catch up* with the environment following requested strategies. It should be noted here that low-level groups can have their own definition of clock $\dot{s}(t)$ as expressions, or in its absence the environment clock $\dot{e}(t)$ is used by default.

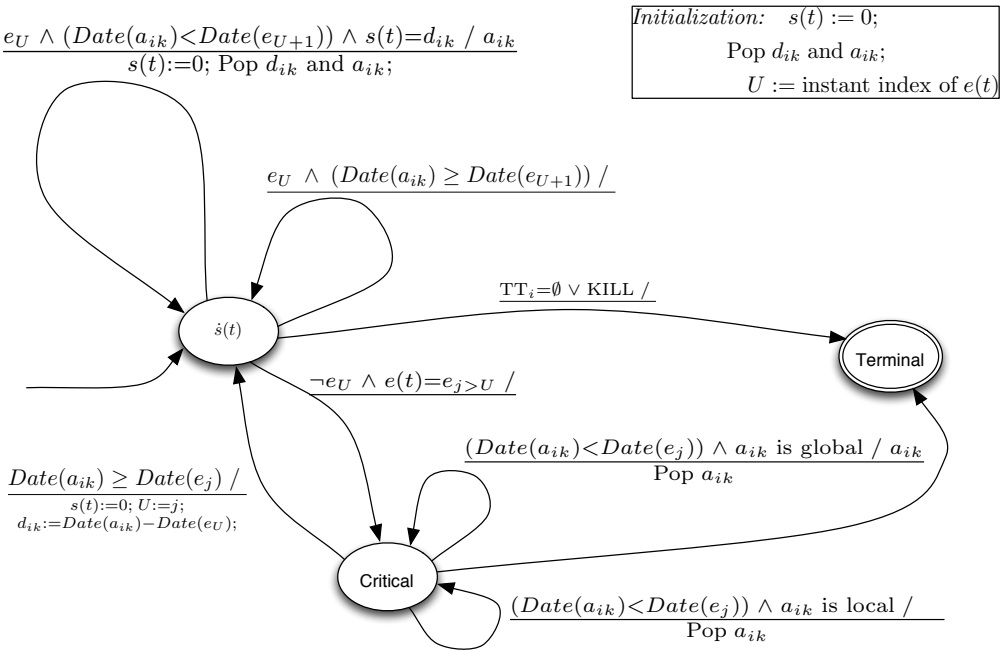


Figure 3.11: Low-level Timer-Trigger runtime semantics for `Tight GFWD`.

Runtime semantics for others types of TTs follow the same guidelines as above and the rest (loop and continuous groups) are in fact derivation of the two classes shown in figures 3.10 and 3.11. Currently, each TT is equipped with its own scheduling mechanism controlled where synchrony is assured by the global clock $\dot{e}(t)$.

During live performance, the scheduling machine assures that event sequences are triggered synchronous to each other and also to the environment, despite sudden changes in environmental dynamics. The *Critical Modes* moreover make the result deterministic in the case of inconsistencies either from the environment or the listening machine. Ordering is preserved, and Antescofo aims to fill in the gap between the authoring and performative aspects of music making for mixed electronic music. Moreover the semantics provided by the language (derived directly from common practices) brings determinacy, fault-tolerance (to certain degree) and more importantly the ability to write things once and as imagined and expect them to happen at runtime without much

compromise.

3.3 Future Perspectives

Antescofo as language and realtime system is relatively young, and its entrance to the field of realtime modeling and computing still younger. The synergy in its use and for its expansion is demanding to explicitly address several issues and creating new research perspectives. Here we briefly introduce several directions already in place or for the few years to come:

Language Developments. The core language of Antescofo is always under development thanks to continuous feedback from composers at Ircam and outside. Recently, *variables* along runtime expressions have been added to the language (Echeveste 2011), making timing variables even more dynamic. Some users have already started making patterns using macros and asking for possibilities of programming dynamic and complex patterns. Whereas synchronous languages are mostly static, recent attempts have shown their power in interactive programming for reactive systems (Mandel et al. 2008). A recent Masters thesis explores such possibilities for Antescofo (Baudart 2012). A natural continuity on this line would then be to define and use dynamic *processes* in the language.

Robustness and Fault-Tolerance. The *Critical Mode* of the scheduling machine described above resembles a *Best Effort Strategy* for fault-tolerance. User level constructs to this end are still minimal and providing common constructs in similar languages such as Giotto (Henzinger, Horowitz, et al. 2003) would not be immediately intuitive for artists. We plan to study various models developed in the literature to this end, test and analyze their performance in known musical situations in order to propose more robust runtime features for fault-tolerance.

Runtime Optimization and Hybrid Scheduling. The hierarchical and tree structures of the score in Antescofo provide a unique and highly structured view of mixed scores. However the fact that each TT has its own scheduler consumes resources that might not be necessary and at times downgrades performance. This issue is even more important given the linear, static and non-concurrent scheduling in Max and PureData. We hope to study more modern scheduling techniques, especially for Globally-asynchronous Locally-Synchronous (GALS) systems (Muttersbach et al. 2000; Krstic et al. 2007) and bring their benefits to online music computing and programming.

Model Checking and Verification. Antescofo can be qualified as a critical system. A failure during a musical performance can lead to artistic disasters. Our aim is to be able to ensure formally that the execution of a program will satisfy expected properties, and on the other hand to provide assistance to composers during authorship. A recent Masters in our team

is studying these procedures (Franchon 2012) following the semantics discussed above and employing linear constraint solvers on timed automata (André et al. 2009).

Probably the most ambitious research horizon that has been opened up by Antescofo is related to the possibilities offered by coupling listening machines to realtime programing environments. To this end, Antescofo is just the very beginning of a new project that aims at more rigorously studying such couplings and their outcome in artistic programing, engineering and production. This is the topic of the next chapter.

IV TOWARDS CYBER-PHYSICAL MUSIC SYSTEMS

The relative success of Antescofo, both with regards to excellence in research and its wide employment by the community, to the eyes of its author is due to the following factors which we believe should be further enhanced in future research:

Human centered computing: Through strong coupling of a machine listening module (chapter 2) and synchronous language (chapter 3) Antescofo brings humans in the loop of computing. Continuous two-way interactions between computers and human musicians is probably the most important reason in rapid adoption of the system, which has thrusted Antescofo from its known scientific goals to expected musical behavior. Further studies should foster this strong and dynamic coupling and beyond existing paradigms exposed in chapter 2.

Coupling of Actions and Perceptions: The joint dynamics of Antescofo's listening machine with its synchronous language has enabled authors to couple actions and perceptions in their programming, whereas previously the two components have been considered disjoint. This coupling has further (and by coincidence) pushed the coordination language of Antescofo to be deployed for *authoring of time and interaction* in real-time applications. The language components of Antescofo with regards to construction of time and interaction are only at its beginning.

Reliability despite fallible components: Individual components of the machine listening in Antescofo are too fallible by themselves, and the reactive language alone is just an intelligent sequencer. However, the joint venture taken by the combination of all components provide reliable results despite many variations and errors during real-time performance. Antescofo's behavior is predictable and deterministic despite non-deterministic nature of the outside sonic environment. A great challenge would be to extend this reliability beyond the current "control" context of the Antescofo engine and to the realm of general and heterogeneous computing.

This chapter is dedicated to our current and prospective research, aiming to extent the pros of Antescofo beyond its currently limited use. To this end, Antescofo is no longer a mere interactive music system but a system at the intersection and union of cyber world (computing) and the physical world (humans). Such study requires revisiting low-level concepts in computer music and interactive arts such as audio engines, distributed computing, scheduling, with regards to continuous interactions with the physical world; and a thorough study of *system behavior* in order to render complex and heterogeneous architectures such as those seen in interactive arts *predictable*. Bridging the mismatch in semantics between physical and software worlds is the goal of the emerging theories in Cyber-Physical Systems. In this chapter, we study scientific challenges for extending Antescofo to a *Cyber-Physical Music System* that brings in physical and cyber worlds, bridges the gap between different practices in digital art, and providing insightful views in designing novel systems in computer music.

4.1 Motivations and Context

A Cyber-Physical System (CPS) is a system featuring a tight integration of, and coordination between heterogeneous devices including physical elements through networks. In contrast to embedded systems with emphasis on computational elements, CPS design is intensely focused on the explicit link between computational and physical elements usually with feedback loops where the environmental processes affect computation and vice versa (E. A. Lee 2010). In this sense, CPS is about *intersection*, not the union, of cyber and physical worlds. *Time* is central in design of CPS and is considered not as an issue of performance but correctness. Thus, the passage of time becomes a central feature of system behavior (Eidson et al. 2012). Physical processes in a system's environment are compositions of many parallel processes and the physical world is not entirely predictable (E. A. Lee 2008). This is in contrast to traditional software processes deeply rooted in sequential steps with few explicit considerations for subsystem failures due to unexpected conditions in the environment.

In this context, a Cyber-Physical Music System is considered as an integration of heterogeneous computational elements (software processes, hardware processors, networks, data acquisition and measurement devices) with that of physical elements. The physical world in this context is comprised of human musicians whose dynamics are provided by machine listening systems acting on (heterogeneous) real- time inputs from the physical environment. The computational elements are comprised of (heterogeneous) models of sound and music computing, off-the-shelf operating systems for real-time audio, as well as hardware and drivers for audio/data delivery and acquisition to and from the physical world. The design of Cyber-Physical Music Systems requires understanding of the joint dynamics of all components. The revival of mentioned literature from a CPS perspective requires revisiting fundamental concepts in each domain to be able to compose *concurrent* systems that deal with dynamically controlling the physical environment by *orchestrating actions* that influence the process. Design and composition of such heterogeneous systems should lead to *predictable*

and *reliable* system behavior, despite non-deterministic components and unpredictability of the physical environment. It is this joint study of dynamics of the Cyber and Physical worlds that sets this approach apart from interactive music systems, by bringing humans in the computation loop via machine listening, and by providing integrated system-level solutions, while contributing to each respective field.

Proposing *Cyber-Physical Music Systems* requires thorough review of fundamentals in both machine listening and real-time sound and music computing, and adapting them to the more realistic views set forth by CPS design. To this respect, the Antescofo system, can be considered a preliminary and limited Cyber-Physical Music System through joint modeling of machine listening and synchronous-reactive orchestration of actions. We submit that its expansion to the wider domain of sound and music computing requires an ambitious research program whose objectives are far from being reached.

4.2 Practical Challenges

4.2.1 Domain-Specific Software and Functional Interoperability

Computer music literature in its practice and research has given birth to various paradigms connecting the physical world of sounds to that of computers. Each approach and paradigm employs specific literature and commonly has led to domain-specific software for each practice. Some of the commonly used softwares include OpenMusic (Assayag, Rueda, et al. 1999) or PWGL (Laurson et al. 2009) for CAO; digital waveguides (Smith 1992) or Modalys (Eckel et al. 1995) for physical modeling of sound synthesis; PureData (Miller Puckette 1997) or Max (Cycling74 2012) for realtime event and signal processing; SuperCollider (McCartney 1996) or CSound (Vercoe 1993) for sound synthesis; Faust (Orlarey et al. 2009) for DSP programming; ChucK (G. Wang 2009), SuperCollider, or Impromptu (A. Sorensen 2005) for live coding; just to name a few.

A piece of interactive computer music neither represents a certain model or technique, nor a workflow specific to a single software. A typical and even simple piece of computer music calls for various computational paradigms from DSP programing mixed with event processing to sound and gesture processing, physical modeling and algorithmic composition.

The practice of authoring and performing computer music is thus at the *intersection*, and not *union*, of different paradigms and approaches. Practicing computer music calls more and more for *functional interoperability* between paradigms and softwares. Some softwares such as PureData and Max have been more or less successful in integrating different approaches and creating such unions. For example, physical modeling of sound using Modalys (solving differential equations) is now an object inside Max despite the fact that the models of computations in the two domains are strictly different. This union causes timing and control problems for the user, not to mention computational inefficiency of this approach. A more efficient approach would be to address this union using *Hybrid Models* (E. A. Lee and Seshia 2011, Chapter 4) but this is practically impossible in Max.

With constant evolution of hardware and software architectures, we also need to consider and add the problem of *preservation* of computer music pieces to our stack. Piece preservation in this context is different from object preservation especially considering realtime contexts. Preservation is different from performance and requires abstraction of processes to study their reliability and predictability. Unfortunately, there is no literature for considering systems *with* physical processes integrated for preservation. This topic requires formal studies and is discussed in section 4.3.3.

Our main argument for future research is that this highly needed interoperability should not be addressed on the *functional* level, but on the *semantic* level of paradigms and their computational models. Such an approach would bring in a CPS design process and make the workflow more explicit for designers, with the drawback of having to redefine formally existing models.

4.2.2 Realtime Computing

Music performance is inherently realtime. Whether an author employs offline processing or realtime algorithms in a piece, the final result is delivered in realtime and possibly interacting with an environment. Existing realtime environments such as PureData and Max focus on performance with weak data structures provided in the language, whereas environments such as OpenMusic focus on rich data structures with no considerations on incremental delivery and performance. This inconsistency has led to a divide in the community for practicing realtime computation (Miller Puckette 2004) with historical criticisms of the practice of realtime from well known composers (Risset 1999; Stroppa 1999) and researchers (Desain et al. 1993) on the paradigm. In the recent years scientists and musicians have attempted to enrich the data structures in Max for realtime computing such as in (Schnell et al. 2005), the *Bach Project* (Agostini et al. 2012) and the new *data structures* in PureData (M. Puckette 2002). These additions consist of appending new structures on top of the existing realtime engines of Max and PureData.

However, in realtime systems the correctness of a system depends not only on the logical results of the computation but also on the time at which the results are produced. One of the primary design objectives of these systems is to support *temporally predictable* execution of computing tasks so that it is guaranteed that there will be timely interactions between tasks and the physical environment, and optimizing performance.

Work on a more rigorous approach to this aspect of realtime systems has followed two largely distinct paths (Burns et al. 2009, Ch. 9): One direction of development has concerned the use of *formally defined language semantics* and timing requirements, together with notations and logic that enable temporal properties to be represented and analyzed (e.g. Kopetz 2011). The other direction has focused on *performance* of realtime systems in terms of feasibility of scheduling the required workload on the available resources (Buttazzo 2005).

Current computer music realtime systems such as Max, PureData and SuperCollider belong to the second category with focus on performance and feasibility. However, even within this domain, employed scheduling algorithms are

based on non-concurrent preemptive scheduling of mixed control and signal processing whereas physical processes (with which they tempt to interact) are all concurrent. Moreover, in all three systems initial scheduling hypotheses lead to arbitrary delays (for Max and PureData) (Miller Puckette 1991) and arbitrary missing cycles for SuperCollider and Impromptu (Andrew Sorensen et al. 2010), therefore destroying the dream for temporal predictability. Such *accepted* system behavior is simply *unacceptable* today: It evades scaling program behavior to the system level, and elude any attempts for behavior predictability and system preservation.

Our argument is that time is a semantic property and not an accident of implementation. Thorough study of joint dynamics between heterogeneous computational models and their temporal semantics is necessary to adopt best strategies. This necessitates constructing a literature on realtime multimedia scheduling which is currently sparse. It is worthy to mention that classical *Multimedia Systems* (Steinmetz et al. 2004) (the root of most realtime systems mentioned above) focus more on timely delivery than timely computation. We believe that a thorough review of realtime computing for continuous multimedia signals is more than necessary. On the other hand, formal approaches help predictability and correctness of realtime programs useful not only for performance but also in the preservation of pieces. Finally, the CPS approach attempts to take benefits of both ends where schedulers are back-ends of formal model abstractions whose timing constraints are non-deterministic (E. A. Lee and Seshia 2011).

4.2.3 Distributed Deployment

Most mixed music programs are deployed over a personal computer that takes numerous inputs (microphone inputs from an orchestra, gesture acquisition inputs, etc) and realtime outputs (e.g. to numerous loudspeakers). Whereas the techniques and models of computations employed in these programs have significantly evolved in the past 10 years, the software architectures handling these transactions have been stagnating as shown in section 3.1. To this one must add the general tendency in computing within recent years to distributed architectures (multi-core PCs) and nomad distribution of computing (tablet devices, embedded and domain-specific computing devices, etc.).

Common architectures shown in section 3.1 have already started exposing their limits even for centralized computing pieces. This was particularly the case of `Tensio` for string quartet and live electronics composed in 2010 by Philippe Manoury with Gilbert Nouno as computer designer. `Tensio` demanded much more concurrent computing than could be handled by the Max scheduler. Before `Tensio`, most composers and computer designers were constrained by CPU power. Ironically this was not the case for `Tensio`. Concurrent modules were designed to occupy multiple non-communicating threads (using `Poly~` objects in Max), but the overall program (following similar architectures as in section 3.1.2) would lead to extreme timing inconsistencies in Max. The developers were finally forced to chop the program to several on the same machine, communicating through Input/Outputs instead of sharing mid-level computations. This leads to considerable design and development efforts forced on the designers.

Unfortunately, the case of `Tensio` is not isolated and many pieces since have encountered the same issue.

The `Tensio` dilemma can be resolved if we take into account correctness on top of performance in scheduling. Technically speaking, in this case the *Utilization* of CPU is low because of poor scheduling strategies, meaning that the scheduler despite having many tasks, is left with many empty spaces with task explosions from time to time leading to temporal inconsistencies. A smarter and adaptive scheduling strategy could have solved the issue. Unfortunately within existing practices no such semantics or syntax is provided to designers.

The `Tensio` dilemma is also present in other forms and practices of digital art, namely for embedded digital art systems such as instrument design or collaborative ensembles. Such practices, despite being rather new, have gained tremendous attention among the digital art communities and beyond, largely thanks to the open-source electronics prototyping platform Arduino (Banzi 2011). Arduino is based on discrete control signals but despite this limitation it has gained attention among gadget and bot makers (Karvinen et al. 2011) and for controlling audio and video software (Noble 2009). Open hardware architectures such as Arduino have opened new possibilities for nomad computing. To this one must add the flow of tablet devices with much more computing power than initially expected. Such practices will bring new possibilities and also more frustration for coordinating distributed computing and their timing constraints.

Foundations of realtime computing in CPS aim to address difficulties discussed above. Our research program aims at providing solutions to these problems by proposing semantics of communication in concurrent (and possibly distributed) systems employing adapted scheduling strategies.

4.3 Scientific Challenges

4.3.1 Heterogeneous Models of Time & Computation

When programing mixed music or interactive art pieces, the overall program is typically made up of distinct and heterogeneous technologies, with their respective literatures and computing paradigms, put together to provide desired internal interactions or with the physical world. Figure 4.1 shows a diagram of some of the most used technologies along their computational paradigms. The software architecture is typically a concurrent integration of several paradigms and the program (or score) should have semantic capabilities for controlling such technologies.

The heterogeneity of models of computations and their temporal natures in computer music make them difficult to study in concurrent architectures and make their integration difficult for designers. A thorough study of computational models in figure 4.1 is outside the scope of any integrated research. However, the semantics of concurrency with regards to their computing structure can be studied, and paves the way for further analysis and deployment in CPSs. Following E. A. Lee and Seshia 2011, we aim at establishing the semantics of concurrent compositions of computer music systems, governed by three sets of rules that we collectively call *Models of Computation (MoC)*. A MoC aims at:

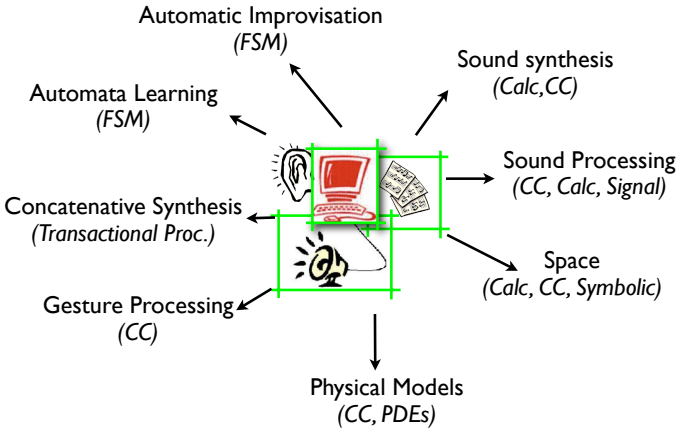


Figure 4.1: Heterogeneous Models of Computations (MoC) in Computer Music.

- Defining what constitute a component;
- Gives semantic to concurrency in the model;
- Defines communication semantics.

For example, *Physical Modeling* of sound synthesis using modal techniques as in (Eckel et al. 1995) are typically continuous systems solving partial differential equations. This is a typical continuous system that can be modeled with discrete abstractions such as timed automata (Alur et al. 1994), synchronous-reactive model, continuous-time model (E. A. Lee and Zheng 2007) or Discrete-Event models (Zeigler et al. 2000). Each approach provides strong semantics with important consequences when embedding the technology in a physical process or next to other models. Among existing views on heterogeneous design of CPS, Ptolemy software stands out with many advantages for computer music mostly due to its *Actor Design* approach (Eker et al. 2003).

Independent study of Models of Computation for computer music paradigm is of little interest in practice since each MoC should in term be combined with other models or delivery services such as audio engines. For example, physical models of sound synthesis should deliver audio to an audio engine where the former can be *Continuous Time Model* and the latter a *Synchronous Dataflow*. This approach is in-line with CPS design procedure, where explicit communication semantics are established between models with the aim of predictability and reliability of results.

In this work, we aim to study, formalize and introduce common computer music MoCs. The challenges we are faced with include: (1) Defining expressive MoCs with strong formal properties, and (2) Define which MoCs is needed for specific application and educate the community of designers to use them.

4.3.2 Temporal Semantics

In Cyber-Physical systems, the passage of time becomes a central feature of system behavior. In fact, it is one of the important constraints distinguishing these

systems from distributed computing in general. Time is central to predicting, measuring and controlling properties of the physical world. In today's programming world, given the source code, the program's initial state, and the amount of time elapsed, we can not reliably predict future program states. When such programs are integrated into a system with environmental dynamics, this makes principle design of the system difficult and the disparity between dynamics of the environment and the program potentially leads to errors which can be potentially catastrophic.

Our goal here is to extend the temporal semantics of Antescofo for the integration of MoCs and their deployment within CPS and embedded architectures. As discussed in section 3.2 the current language and system have their roots in Reactive Synchronous systems (Halbwachs 1993) where the duration of computational processes is assumed zero or within a clock tick. In the current message-passing system of Antescofo this is a feasible hypothesis. However, if the action semantics are to be augmented by processes representing MoCs, further considerations should be made.

Existing approaches, drawn from control theory, assume an oracle clock available simultaneously in all parts of the system. Below, we summarize several important candidates for our studies.

Synchronous-Reactive Approach. The synchronous-reactive (SR) framework is based on a hypothetical zero-time computation assumption of actions (G. Berry et al. 1985) which provide powerful semantics for predictable and critical systems. SR framework can be extended to communication semantics as opposed to language semantics for coordinating MoCs with heterogeneous natures. This is for example the approach taken in (E. A. Lee and Zheng 2007) for Ptolemy where synchronous-reactive coordination frameworks are used within continuous-time models and discrete-event systems altogether. The notion of *time* in SR is rather implicit and can be extended to any data-flow sequence. Recent works have added *polymorphic clock calculus* to SR frameworks (Caspi et al. 1996). SR systems have been successfully applied to critical realtime systems with strict safety constraints (Gérard Berry 2003). A relaxed version of the clock calculus has been applied to embedded media devices such as HDTVs (Cohen et al. 2005).

Time-Triggered Architectures. Another approach consists of leveraging the zero-time computation of synchronous-reactive framework by explicitly considering computation times for action nodes as proposed in Time-Triggered Architectures (TTA) (Kopetz and Bauer 2003). TTA provides a computing infrastructure for the design and implementation of dependable distributed embedded systems. TTA requires a sparse global time of known precision present in every component and explicitly considers communication nodes to guarantee fault-tolerance. This global clock has to be uniform and the analysis of the dynamics has to include the imperfections (which is not always known). TTA can be clearly distinguished from event-triggered systems in both theory and practice. For example

TTA systems usually require a single interrupt timer for the global process. TTA has been successfully applied to industrial applications mostly in the automotive industry. It has also incarnated as a dedicated language for embedded realtime programming (Henzinger, Horowitz, et al. 2003).

Discrete-Event Systems. Discrete-event systems (DE) have been used for decades as a way to build simulations for variety of applications due to pioneering formalisms in (Zeigler et al. 2000). The framework endows state-machines (more explicitly Moore machines) with a non-zero life-span for each state (as opposed to simple reactions). The key idea is to associate to each event a *time stamp* within a model of time that ought to be comparable. A DE actor reacts to time-stamped events in order and employs event queues. Operational semantics of DE systems have given birth to many variants such as combining them with an SR model of computation. A recently proposed strategy called *PTIDES* for programming temporally integrated distributed embedded systems, leverages this approach for CPS by considering *model time* versus *physical time* in runtime and schedulability analysis. Another interesting design factor in *PTIDES* is its independence from platform-specific system timers (Eidson et al. 2012).

4.3.3 Preservation & Predictablity

A computer music program can be considered as a collection of programs and technologies embedded into an integral system as shown in section 3.1.2. These programs are characterized by their interactions with the physical world and themselves in realtime. Similar to embedded system design, non-functional requirements of computer music systems can be classified as follows (Henzinger and Sifakis 2007):

Reaction requirements, which concern the interaction of the system with the environment, and

Execution requirements, which concerns the interaction of the system with the platform (software and hardware components).

In computer music, one important source of *reaction requirement* is user expectation from music scores or musical specifications shown in section 3.1.1. These authored specification typically give rise to other reaction requirements such as response time, which bounds the worst or average-case delay between an external stimulus of the system and its response.

An important source of execution requirement is resource constraints which may be imposed by hardware or software, audio service constraints, controller hardware delays, use of specific scheduling algorithms or communication protocols that are all typical in any computer music performance.

By contrast, a system that is not embedded has only functional requirements. Digital Audio Effects, when considered alone, can be viewed as functional programs where there is neither a reaction requirement (it is not specified when the effect output must be provided), nor an execution requirement (it is not

specified how much memory or resources the effect may consume). Reaction and execution requirements are absent not because there no corresponding user expectations or resource constraints (in practice a realtime audio effect has time-sensitive delivery constraints), but these constraints are neglected because they are secondary to functional requirements.

It is not prudent at all to abstract response time and resource utilization for computer music systems whereas locally such abstractions are less harmful. A computer music piece undergoes time requirements inferring resource, computational and variants of time constraints among modules and upon stimuli arrivals. Moreover, a computer music program should inherently be an *abstraction refinement* of its score; permitting the realization of the same piece on other platforms and with the (constraint) evolution of computing platforms. This last issue, of utmost importance in numerical art, is strongly related to *preservation*. This domain of study is relatively new. To our knowledge, the only systematic view on preservation is the work in (Barkati et al. 2012) using functional formalisms whose limitations we brought up previously.

For this project, we argue that the grand challenge in computer music systems and design is to identify high-level programming models that expose the reaction and execution properties of a system in a way that,

1. permits the programmer to express desired reaction and execution requirements, and

2. permits the compiler and/or run-time system to ensure that these requirements are satisfied.

While there have been many proposals and languages for realtime programming in computer music with formal models (such as those discussed in section 4.2.1), we believe that none has attained both objectives. For example, the implicit specification of timing properties in Max and PureData address objective (2) (at least for small programs), but have failed with respect to (1). As a result, computer action specifications are either trivial or programmed as a result of many tests and trial-and-errors.

We submit that a successful solution to these challenges has to exhibit two key characteristics: First, the programming models must have the property that all software written in the model be *predictable*, not only on the functional level but also in its reaction and execution properties; and second, the programming model must have the property that all software written in the model be *robust*, in the sense that its reaction properties change only slightly if the environment changes slightly. This last property is crucial for art-code preservation.

Predictability can be achieved through *determinism*. One approach to achieve predictable systems is to build them entirely from deterministic parts. There are two problems with this approach: First, they are not realistic as many basic interactive components can be non-deterministic; and second, they usually do not provide generally acceptable solutions. The key question is not how we can build complex programs out of deterministic parts, but how we can build deterministic abstraction layers from non-deterministic components. To approach this question, one needs to study various sources of non-determinism in computer

music systems such as *input non-determinism*, *unobservable and observation implementation non-determinism*, and *don't care situations* (Henzinger 2008). To this, one must add time-determinism required by computer music CPS. Scheduled systems (dominant in computer music today) are often unpredictable. A particularly successful attempt for this kind of observable non-determinism is the synchronous programming languages (Gérard Berry 2003; Halbwachs 1993). Another alternative approach for time-determinism is the *Giotto* language approach (Henzinger, Horowitz, et al. 2003).

The second challenge deals with constructing systems whose behavior is robust in the presence of perturbation. The difficulty here is due to the fact that in computer science, unlike other engineering disciplines, we often lose sight of the fact that a mathematical representation of a software system is just a model, and that the actual system is physical, executing on a physical imperfect platform and interacting with a physical unknowable environment. This issue deals with heterogenous natures of computation and time in model abstractions and the challenge of building, on top of non-continuous system implementations, system abstractions that are continuous with regards to physical quantities (E. A. Lee 2009). This can be achieved, in terms, by a careful study of computer music MoCs and their formal operational and communication semantics as proposed in section 4.3.1.

Our goal in this project is thus to *link system behavior to implementation*, by leveraging computer music practices that explicitly inform the determinism of system abstractions, and through such formalisms provide *Semantic-Preserving Translations and Implementations*, paving the way for preservation of computer music programs and systems.

4.3.4 Architectures for Embedded Computer Audition

An important challenge in our definition of Cyber-Physical Music Systems is to leverage interactions with the environment and bring such considerations into the design process. Antescofo, seen as a preliminary CPS, provides such interactions through machine listening limited to realtime alignment of audio to scores and enabling formal descriptions of actions and reactions.

One of the main goals of this project is to bring machine listening capabilities to the next step, by augmenting it with state-of-the-art machine listening methods and other forms of listening than those currently available. This part of our project is concerned as much with machine learning and signal processing (Chapter 2) as software architectures, the idea being the capability of rapid and easy deployment of machine listening techniques to integrated CPS and embedded systems.

A similar ongoing attempt in the field of computer vision is the OpenCV Project[1] for Open Source Computer Vision Library (Bradski 2002). OpenCV is a library of programming functions aimed at realtime computer vision, free under the open source BSD license and cross-platform. Today, OpenCV gathers millions of users from artists to robotic industries together and has drawn

[1] http://www.opencv.org/

standards for such techniques especially for embedded vision protocols (Bradski and Kaehler 2008).

Our goal is to collect computer audition methods within an open architecture framework similar to OpenCV. This project requires support from the community and will nurture both aspects of our project. Its outcome will lead to further extensions of the Antescofo coordination language and interactions. To our knowledge and to this date, no similar ambitions exist in realtime computer audition.

BIBLIOGRAPHY

Abdallah, Samer M. and Mark D. Plumbley (2004). "Polyphonic transcription by non-negative sparse coding of power spectra." In: *ISMIR*. URL: http://ismir2004.ismir.net/proceedings/p058-page-318-paper216.pdf.

Agostini, Andrea and Daniele Ghisi (May 2012). "GESTURES, EVENTS AND SYMBOLS IN THE BACH ENVIRONMENT." In: *Journées d'Informatique Musicale*.

Alur, Rajeev and David L. Dill (April 1994). "A theory of timed automata." In: 126 (2), pp. 183–235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8. URL: http://portal.acm.org/citation.cfm?id=180782.180519.

Amari, S. and H. Nagaoka (2000). *Methods of information geometry*. Vol. 191. Translations of mathematical monographs. Oxford University Press.

André, Étienne et al. (October 2009). "An Inverse Method for Parametric Timed Automata." In: 20.5, pp. 819–836. DOI: 10.1142/S0129054109006905. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/ACEF-ijfcs09.pdf.

Arulampalam, S. et al. (February 2002). "A Tutorial on Particle Filters for Online Non-linear/Non-Gaussian Bayesian Tracking." In: 50.2, pp. 174–188. URL: http://citeseer.ist.psu.edu/article/arulampalam01tutorial.html.

Assayag, Gerard and Andrew Gerzso, eds. (June 2009). *New Computational Paradigms for Computer Music*. Sciences de la musique. Editions Delatour.

Assayag, Gerard, Camilo Rueda, et al. (1999). "Computer Assisted Composition at Ircam: From PatchWork to OpenMusic." In: 23.3. URL: http://www.ircam.fr/equipes/repmus/RMPapers/CMJ98/index.html.

Banerjee, Arindam et al. (2005). "Clustering with Bregman Divergences." In: 6, pp. 1705–1749. ISSN: 1533-7928.

Banzi, Massimo (September 2011). *Getting Started with arduino*. 2nd ed. O'Reilly Media / Make.

Barbaresco, Frederic (2009). "Applications of Information Geometry to Radar Signal Processing." In: *Emerging Trends in Visual Computing, LIX Fall Colloquium*. Ed. by Frank Nielsen. Vol. 5416. Lecture Notes in Computer Science. Springer. ISBN: 978-3-642-00825-2.

Barkati, K., Y. Orlarey, and J. Barthélemy (2012). "Abstraction du processus temps réel: une stratégie pour la préservation à long terme." In:

Basseville, Michèle and Igor V. Nikiforov (1993). *Detection of abrupt changes: theory and application*. Prentice-Hall, Inc. ISBN: 0-13-126780-9.

Baudart, Guillaume (September 2012). *Antescofo: Vers une programmation synchrone*.

Berry, G. and L. Cosserat (1985). "The ESTEREL synchronous programming language and its mathematical semantics." In: *Seminar on Concurrency*. Springer, pp. 389–448.

Berry, Gérard (2003). *The Effectiveness of Synchronous Languages for the Development of Safety-Critical Systems*. URL: http://www.esterel-technologies.com/DO-178B/files/The-Effectiveness-of-Synchronous-Languages-for-the-Development-of-Safety-Critical-Systems.pdf.

Boulez, Pierre and Andrew Gerzso (1988). "Computers in Music." In: 258.4, pp. 44–50. URL: http://articles.ircam.fr/textes/Boulez88c/.

Bradski, G. (2002). "OpenCV: Examples of use and new applications in stereo, recognition and tracking." In: *Proc. Intern. Conf. on Vision Interface (VI'2002)*, p. 347.

Bradski, G. and A. Kaehler (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media.

Burago, D., Y. Burago, and S. Ivanov (2001). *A course in metric geometry*. Vol. 33. Graduate Studies in Mathematics. American Mathematical Society.

Burns, Alan and Andy Wellings (2009). *Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX*. 4th ed. Addison Wesley.

Buttazzo, Giorgio C. (2005). *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*. Springer-Verlag. ISBN: 0387231374.

Caspi, Paul and Marc Pouzet (June 1996). "Synchronous Kahn networks." In: 31.6, pp. 226–238. ISSN: 0362-1340. DOI: 10.1145/232629.232651. URL: http://doi.acm.org/10.1145/232629.232651.

Cauchi, Benjamin et al. (April 2012). "Sparse representations for modeling environmental acoustic scenes, application to train stations soundscapes." In: *CFA - Congres Français d'Acoustique*. URL: http://articles.ircam.fr/textes/Cauchi12a/.

Chabot, Xavier, Roger Dannenberg, and George Bloch (October 1986). "A workstation in live performance: Composed improvisation." In: *International Computer Music Conference (ICMC)*, pp. 537–540.

Chadabe, Joel (1984). "Interactive Composing: An Overview." In: 8.1, pp. 22–27. ISSN: 01489267. URL: http://www.jstor.org/stable/3679894.

Chai, Wei (September 2005). *Automated Analysis of Musical Structure*. URL: http://alumni.media.mit.edu/~chaiwei/papers/whole0622.pdf.

Cichocki, A. et al. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley.

Cohen, Albert et al. (September 2005). "Synchroning Periodic Clocks." In: *ACM International Conference on Embedded Software (EMSOFT'05)*.

Cont, Arshia (2004). *Improvement of Observation Modeling for Score Following*.

— (May 2006a). "Realtime Audio to Score Alignment for Polyphonic Music Instruments Using Sparse Non-negative constraints and Hierarchical HMMs." In: *IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*. Toulouse.
— (September 2006b). "Realtime Multiple Pitch Observation using Sparse Non-negative Constraints." In: *International Symposium on Music Information Retrieval (ISMIR)*. Victoria, CA., pp. 206–2011. URL: http://cosmal.ucsd.edu/arshia/papers/ArshiaCont_ismir2006.pdf.
— (August 2008a). "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." In: *Proceedings of International Computer Music Conference (ICMC)*. Belfast. URL: http://articles.ircam.fr/textes/Cont08a/.
— (September 2008b). *Modeling Musical Anticipation: From the time of music to the music of time*.
— (May 2010). "A coupled duration-focused architecture for realtime music to score alignment." In: 32.6, pp. 974–987. URL: http://articles.ircam.fr/textes/Cont09a/.
— (July 2011). "On the creative use of score following and its impact on research." In: *Sound and Music Computing*. URL: http://articles.ircam.fr/textes/Cont11a/.
— (June 2012). "L'ordinateur qui joue comme un musicien." In: 465, pp. 68–72.
Cont, Arshia, Shlomo Dubnov, and Gerard Assayag (May 2011). "On the Information Geometry of Audio Streams with Applications to Similarity Computing." In: 19.4. URL: http://articles.ircam.fr/textes/Cont10b/index.pdf.
Cont, Arshia, Shlomo Dubnov, and David Wessel (September 2007). "Realtime Multiple-pitch and Multiple-instrument Recognition For Music Signals using Sparse Non-negative Constraints." In: *Proceedings of Digital Audio Effects Conference (DAFx)*. Bordeaux.
Cont, Arshia, José Echeveste, et al. (September 2012). "Correct Automatic Accompaniment Despite Machine Listening or Human Errors in Antescofo." In: *International Computer Music Conference (ICMC)*. URL: http://hal.inria.fr/hal-00718854/PDF/Antescofo_ICMC2012_new.pdf.
Cont, Arshia, Diemo Schwarz, and Norbert Schnell (2004). "Training Ircam's Score Follower." In: *AAAI Fall Symposium on Style and Meaning in Language, Art and Music*. URL: http://cosmal.ucsd.edu/arshia/papers/AAAI04/.
— (March 2005). "Training Ircam's Score Follower." In: *IEEE International Conference on Acoustics and Speech Signal Processing (ICASSP)*. Philadelphia. URL: http://cosmal.ucsd.edu/arshia/papers/ICASSP05/.
Cont, Arshia, Diemo Schwarz, Norbert Schnell, and Christopher Raphael (September 2007). "Evaluation of Real-Time Audio-to-Score Alignment." In: *International Symposium on Music Information Retrieval (ISMIR)*. Vienna, Austria.
Cuvillier, Philippe (September 2012). *Suivi de partition: étude du cadre multi-objets pour l'inférence de position*.

Cycling74 (2012). *Max/MSP realtime graphical programming environment*. URL: http://www.cycling74.com/.

Dannenberg, Roger B. (1984). "An On-Line Algorithm for Real-Time Accompaniment." In: *Proceedings of the International Computer Music Conference (ICMC)*, pp. 193–198.

Desain, P. and H. Honing (1993). "The Mins of max." In: 17.2, pp. 3–11.

Dessein, Arnaud (2009). *Incremental multi-source recognition with non-negative matrix factorization*. Master. URL: http://articles.ircam.fr/textes/Dessein09b/index.pdf.

— (December 2012). *Computational Methods of Information Geometry with Real-Time Applications in Audio Signal Processing*. URL: http://hal.inria.fr/tel-00768524/PDF/Dessein2012PhD.pdf.

Dessein, Arnaud and Arshia Cont (September 2011). "Segmentation statistique de flux audio en temps-réel dans le cadre de la géométrie de l'information." In: *GRETSI Special Session on Information Geometry*.

Dessein, Arnaud, Arshia Cont, and Guillaume Lemaitre (August 2010). "Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence." In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*.

— (2013). "Real-Time Detection of Overlapping Sound Events with Non-Negative Matrix Factorization." In: *Matrix Information Geometry*. Ed. by Frank Nielsen and Rajendra Bhatia. Springer Berlin Heidelberg, pp. 341–371. ISBN: 978-3-642-30232-9. URL: http://dx.doi.org/10.1007/978-3-642-30232-9_14.

Donin, Nicolas (2012). "The Act of Musical Composition – Studies in the Creative Process," in: ed. by Dave Collins. Ashgate. Chap. Empirical and Historical Musicologies of Compositional Processes: Towards a Cross-fertilization.

Dubnov, Shlomo and Xavier Rodet (January 2003). "Investigation of phase coupling phenomena in sustained portion of musical instruments sound." In: 113, pp. 348–359.

Echeveste, José (2011). *Stratégies de synchronisation et gestion des variables pour l'accompagnement musical automatique*. Master. URL: http://articles.ircam.fr/textes/Echeveste11a/index.pdf.

Echeveste, José et al. (November 2011). "Formalisation des relations temporelles entre une partition et une performance musicale dans un contexte d'accompagnement automatique." In: *Colloque Modélisation des Systèmes Réactifs (MSR)*.

— (2012). "Antescofo: A Domain Specific Language for Realtime Musician-Computer Interaction." In: (Submitted).

Eckel, G., F. Iovino, and R. Caussé (1995). "Sound synthesis by physical modelling with Modalys." In: *Proceedings of the International Symposium of Music Acoustics*.

Eidson, J.C. et al. (January 2012). "Distributed Real-Time Software for Cyber-Physical Systems." In: 100.1, pp. 45–59. ISSN: 0018-9219. DOI: 10.1109/JPROC.2011.2161237.

Eker, J. et al. (January 2003). "Taming heterogeneity - the Ptolemy approach." In: 91.1, pp. 127–144. ISSN: 0018-9219. DOI: 10.1109/JPROC.2002.805829.

Foote, Jonathan (March 1997). "A Similarity Measure for Automatic Audio Classification." In: *Proceedings AAAI 1997 Spring Symposium on Intelligent*

Integration and Use of Text, Image, Video and Audio Corpora. American Association for Artificial Intelligence.

Foote, Jonathan and M. Cooper (2003). "Media Segmentation using SelfSimilarity Decomposition." In: *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*. Vol. 5021, pp. 167–175.

Franchon, Lea (2012). *Temporal Analysis of Mixed Intrumental/Electronic Music Scores*.

Grubb, Lorin and Roger B. Dannenberg (1994). "Automating Ensemble Performance." In: *Proceedings of the ICMC*, pp. 63–69.

Halbwachs, Nicolas (1993). *Synchronous Programming of Reactive Systems*. Kluwer Academics.

Henzinger, T.A. (2008). "Two challenges in embedded systems design: predictability and robustness." In: 366.1881, pp. 3727–3736.

Henzinger, T.A., B. Horowitz, and C.M. Kirsch (January 2003). "Giotto: a time-triggered language for embedded programming." In: 91.1, pp. 84–99. ISSN: 0018-9219. DOI: 10.1109/JPROC.2002.805825.

Henzinger, T.A. and J. Sifakis (October 2007). "The Discipline of Embedded Systems Design." In: 40.10, pp. 32–40. ISSN: 0018-9162. DOI: 10.1109/MC.2007.364.

Hoyer, Patrik O. (2004). "Non-negative Matrix Factorization with Sparseness Constraints." In: 5, pp. 1457–1469.

Karvinen, Tero and Kimmo Karvinen (March 2011). *Make: Arduino Bots and Gadgets: Six Embedded Projects with Open Source Hardware and Software*. 2nd ed. O'Reilly Media / Make.

Kawahara, H. et al. (1999). "Fixed point analysis of frequency to instantaneous frequency mapping for accurate estimation of F0 and periodicity." In: *Eurospeech*. Vol. 6, pp. 2781–2784.

Gossip is Philosophy (1995) 3.05. URL: http://www.wired.com/wired/archive/3.05/eno.html.

Kim, Kyoung-Dae and P.R. Kumar (13 2012). "Cyber-Physical Systems: A Perspective at the Centennial." In: 100.13, pp. 1287–1308. ISSN: 0018-9219. DOI: 10.1109/JPROC.2012.2189792.

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Vol. 25. Springer-Verlag New York Inc.

Kopetz, H. and G. Bauer (January 2003). "The time-triggered architecture." In: 91.1, pp. 112–126. ISSN: 0018-9219. DOI: 10.1109/JPROC.2002.805821.

Krstic, M. et al. (2007). "Globally asynchronous, locally synchronous circuits: Overview and outlook." In: 24.5, pp. 430–441.

Large, Edward W. and Marie Riess Jones (1999). "Dynamics of Attending: How People Track Time-Varying Events." In: 106.1, pp. 119–159.

Laurson, M., M. Kuuskankare, and V. Norilo (2009). "An overview of pwgl, a visual programming environment for music." In: 33.1, pp. 19–31.

Lee, Daniel D. and H. Sebastian Seung (2001). "Algorithms for Non-negative Matrix Factorization." In: *Advances in Neural Information Processing Systems 13*. Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp. MIT Press, pp. 556–562.

Lee, Edward A. (May 2008). "Cyber Physical Systems: Design Challenges." In: *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*. Invited Paper. URL: http://chess.eecs.berkeley.edu/pubs/427.html.

— (May 2009). "Computing Needs Time." In: 52.5, pp. 70–79. URL: http://chess.eecs.berkeley.edu/pubs/615.html.

— (June 2010). "CPS Foundations." In: *Proc. of the 47th Design Automation Conference (DAC)*. ACM, pp. 737–742. URL: http://chess.eecs.berkeley.edu/pubs/804.html.

Lee, Edward A. and Sanjit A. Seshia (2011). *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. Lee & Seshia. ISBN: 978-0-557-70857-4. URL: http://leeseshia.org/.

Lee, Edward A. and Haiyang Zheng (2007). "Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems." In: *EMSOFT '07: Proceedings of the 7th ACM & IEEE international conference on Embedded software*. Salzburg, Austria: ACM, pp. 114–123. ISBN: 978-1-59593-825-1. DOI: http://doi.acm.org/10.1145/1289927.1289949.

Logan, B. and S. Chu (2000). "Music summarization using key phrases." In: 2, pp. II749–II752.

Loy, G. and C. Abbott (1985). "Programming languages for computer music synthesis, performance, and composition." In: 17.2, pp. 235–265.

Machover, Tod and J. Chung (1989). "Hyperinstruments: Musically intelligent and interactive performance and creativity systems." In: *International Computer Music Conference (ICMC)*, pp. 186–190.

Mahler, Ronald P. S. (2007). *Statistical multisource-multitarget information fusion*. Artech House. ISBN: 978-1-59693-092-6.

Mallat, Stéphane (November 27, 2011). "Group Invariant Scattering." In: eprint: 1101.2286. URL: http://arxiv.org/abs/1101.2286.

Mandel, Louis and Florence Plateau (April 2008). "Interactive Programming of Reactive Systems." In: *Proceedings of Model-driven High-level Programming of Embedded Systems (SLA++P'08)*. Electronic Notes in Computer Science. Elsevier Science Publishers, pp. 44–59. URL: MandelPlateau-SLAP-2008.pdf.

Manoury, Philippe (1990). *La note et le son*. L'Hamartan.

— (November 2007). "Considérations (toujours actuelles) sur l'état de la musique en temps réel." In: URL: http://etincelle.ircam.fr/prospectives.html.

McCartney, James (1996). "SuperCollider: a new real time synthesis language." In: *Proceedings of the International Computer Music Conference*. URL: http://www.audiosynth.com/icmc96paper.html.

MIREX (August 2006). *Score Following Evaluation Proposal*. URL: http://www.music-ir.org/mirex/wiki/2006:Score_Following_Proposal (visited on 08/2012).

— (August 2010). *Multiple Fundamental Frequency Estimation & Tracking Results*. URL: http://www.music-ir.org/mirex/wiki/2010:Multiple_Fundamental_Frequency_Estimation_%26_Tracking_Results (visited on 08/2012).

Montecchio, Nicola and Arshia Cont (May 2011a). "A Unified Approach to Real Time Audio-to-Score and Audio-to-Audio Alignment Using Sequential Montecarlo Inference Techniques." In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

— (September 2011b). "Accelerating the Mixing Phase in Studio Recording Productions by Automatic Audio Alignemtn." In: *12th International Symposium on Music Information Retrieval (ISMIR)*. in press.

Muttersbach, J., T. Villiger, and W. Fichtner (2000). "Practical design of globally-asynchronous locally-synchronous systems." In: *Advanced Research in Asynchronous Circuits and Systems (ASYNC 2000) Proceedings. Sixth International Symposium on*. IEEE, pp. 52–59.

Nielsen, Frank, ed. (2009). *Emerging Trends in Visual Computing, LIX Fall Colloquium, ETVC 2008, Palaiseau, France, November 18-20, 2008. Revised Invited Papers*. Vol. 5416. Lecture Notes in Computer Science. Springer. ISBN: 978-3-642-00825-2.

Noble, Joshua (2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks*. 1st. O'Reilly Media, Inc. ISBN: 978-0-596-15414-1.

Orio, Nicola and F. Déchelle (2001). "Score Following Using Spectral Analysis and Hidden Markov Models." In: *Proceedings of the ICMC*.

Orlarey, Y., D. Fober, and S. Letz (2009). "Faust: an efficient functional approach to DSP programming." In:

Page, E. S. (1954). "Continuous Inspection Scheme." In: 41, pp. 100–115.

Peeters, Geoffroy (2004). "Deriving Musical Structures from Signal Analysis for Audio Summary Generation: "Sequence" and "State" approach." In: *CMMR*. Vol. 2771.

Pennec, Xavier (2009). "Statistical Computing on Manifolds: From Riemannian Geometry to Computational Anatomy." In: *Emerging Trends in Visual Computing*. Springer-Verlag, pp. 347–386. ISBN: 978-3-642-00825-2. DOI: http://dx.doi.org/10.1007/978-3-642-00826-9_16.

Puckette, M. (September 2002). "Using Pd as a score language." In: *Proc. Int. Computer Music Conf.* pp. 184–187. URL: http://www.crca.ucsd.edu/~msp.

Puckette, Miller (1988). "The Patcher." In: *Proceedings of International Computer Music Conference (ICMC)*, pp. 420–429.

— (1991). "Combining Event and Signal Processing in the MAX Graphical Programming Environment." In: 15, pp. 68–77.

— (September 1997). "Pure data." In: *Proc. Int. Computer Music Conf.* pp. 224–227. URL: http://www.crca.ucsd.edu/~msp.

— (2002). "Max at Seventeen." In: 26.4, pp. 31–43. ISSN: 0148-9267. DOI: http://dx.doi.org/10.1162/014892602320991356.

— (2004). "A divide between 'compositional' and 'performative' aspects of Pd." In: *First International Pd Convention*.

Puckette, Miller and Cort Lippe (1992). "Score Following in Practice." In: *Proceedings of the ICMC*, pp. 182–185.

Raphael, Christopher (1999). "A Probabilistic Expert System for Automatic Musical Accompaniment." In: 10.3, pp. 487–512.

Raphael, Christopher (2001). "Music Plus One: A System for Expressive and Flexible Musical Accompaniment." In: *Proceedings of the ICMC*.

Risset, Jean-Claude (1999). "Composing in real-time?" In: 18.3, pp. 31–39.

Rowe, Robert (1992). *Interactive music systems: machine listening and composing*. MIT Press. ISBN: 0-262-18149-5.

Scheirer, Eric D. (2000). *Music listening systems*. URL: http://web.media.mit.edu/~tristan/Classes/MAS.945/Papers/Technical/Scheirer_Thesis.pdf.

Schnell, Norbert et al. (September 2005). "FTM — Complex data structures for Max." In: *International Computer Music Conference (ICMC)*. URL: http://mediatheque.ircam.fr/articles/textes/Schnell05a/.

Smaragdis, Paris and Judy Brown (2003). "Non-negative matrix factorization for polyphonic music transcription." In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

Smith, J.O. (1992). "Physical modeling using digital waveguides." In: 16.4, pp. 74–91.

Snoussi, Hichem and Cédric Richard (January 2009). "Monte Carlo Tracking on the Riemannian Manifold of Multivariate Normal Distributions." In: *IEEE DSP'09*.

Sorensen, A. (2005). "Impromptu: An interactive programming environment for composition and performance." In: *Proceedings of the Australasian Computer Music Conference 2009*.

Sorensen, Andrew and Henry Gardner (2010). "Programming with time: cyber-physical programming with impromptu." In: *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*. OOPSLA '10. Reno/Tahoe, Nevada, USA: ACM, pp. 822–834. ISBN: 978-1-4503-0203-6. DOI: 10.1145/1869459.1869526. URL: http://doi.acm.org/10.1145/1869459.1869526.

Steinmetz, R. and K. Nahrstedt (2004). *Multimedia systems*. Springer.

Stroppa, Marco (1999). "Live electronics or live music? Towards a critique of interaction." In: 18.3, pp. 41–77.

Sukittanon, Somsak, Les E. Atlas, and James W. Pitton (2004). "Modulation-Scale Analysis for Content Identification." In: 52.10, pp. 3023–3035.

Szendy, Peter (1998). "Musique, temps réel." In: 14. URL: http://articles.ircam.fr/textes/Szendy98b/.

Vercoe, Barry (1984). "The Synthetic Performer in the Context of Live Performance." In: *Proceedings of the ICMC*, pp. 199–200.

— (1993). *Csound, A Manual for the Audio Processing System and Supporting Programs with Tutorials*. 1993.

"Multiple F0 estimation" (2006). In: *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*. Ed. by D.-L. Wang and G.J. Brown. IEEE Press / Wiley, pp. 45–72.

Wang, G. (2009). *The ChucK audio programming language." A strongly-timed and on-the-fly environ/mentality"*.

Xia, G-S. et al. (2012). "Compact Representations of Stationary Dynamic Textures." In: *Proc. ICIP'12*. URL: http://hal.archives-ouvertes.fr/hal-00662719.

Zeigler, B.P., H. Praehofer, and T.G. Kim (2000). *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. 2nd ed. Academic Press.

Zhang, Jun (2004). "Divergence function, duality, and convex analysis." In: 16.1, pp. 159–195. ISSN: 0899-7667. DOI: http://dx.doi.org/10.1162/08997660460734047. URL: http://neco.mitpress.org/cgi/content/full/16/1/159.