

Atti del

X Colloquio di Informatica Musicale

Editors: Goffredo Haus & Isabella Pighi



AIMI
Associazione di Informatica Musicale Italiana

LIM-DSI
Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Davide Rocchesso

Atti del
**X Colloquio di
Informatica Musicale**

Milano, 2-4 dicembre 1993

Editor: Goffredo Haus & Isabella Pighi

AIMI
Associazione di Informatica Musicale Italiana

LIM-DSI
Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

COMITATO SCIENTIFICO

<i>Mario Baroni</i>	(Università di Bologna)
<i>Antonio Camurri</i>	(Università di Genova)
<i>Jacques Chareyron</i>	(Università di Milano)
<i>Giovanni De Poli</i>	(Università di Padova)
<i>Goffredo Haus</i>	(Università di Milano)
<i>Aldo Piccialli</i>	(Università di Napoli)
<i>Sylviane Sapir</i>	(IRIS)

COMITATO MUSICALE

<i>Lelio Camilleri</i>	(Conservatorio di Bologna)
<i>Mauro Graziani</i>	(Università di Padova)
<i>Alessandro Melchiorre</i>	(Civica Scuola di Musica di Milano)
<i>Angelo Paccagnini</i>	(Università di Milano)
<i>Nicola Sani</i>	(RAI-SAT)

COMITATO ORGANIZZATORE

Goffredo Haus; Angelo Paccagnini; Isabella Pighi; Dante Tanzi
LIM-DSI, Università degli Studi di Milano

Il X Colloquio di Informatica Musicale è promosso da:

AIMI, Associazione di Informatica Musicale Italiana
e
LIM-DSI, Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

con il patrocinio di:



IEEE Computer Society
Task Force on Computer Generated Music
&
North Italy Section

con il contributo di:



Consiglio Nazionale delle Ricerche,
Comitato Scienze e Tecnologie
dell'Informazione e Progetto Finalizzato
"Sistemi Informatici e Calcolo Parallelo"



SiliconGraphics
Computer Systems



Civica Scuola di Musica
Comune di Milano

INTERSOFT

INTRODUZIONE

Goffredo Haus

Direttore Scientifico

LIM - Laboratorio di Informatica Musicale

Dipartimento di Scienze dell'Informazione

Università degli Studi di Milano

Dopo l'ormai lontana edizione del 1977, il Colloquio di Informatica Musicale torna all'Università degli Studi di Milano. Molto tempo è passato e molte cose sono cambiate. L'informatica musicale è cresciuta come disciplina scientifica e artistica e gli ambiti applicativi che la caratterizzano si sono sempre più diversificati e specializzati. In Italia i Colloqui di Informatica Musicale hanno assunto un ruolo sempre più efficace nella comunicazione nazionale e internazionale dei più avanzati risultati della ricerca nel campo dell'informatica musicale.

La X edizione del Colloquio è, come accade dal 1981, promossa dall'AIMI - Associazione di Informatica Musicale Italiana ed è organizzata dal LIM - Laboratorio di Informatica Musicale del Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano.

La sede di questa edizione è attiva nel settore dell'informatica musicale dal 1975 ed è un polo di ricerca scientifico-tecnologica principalmente dedicato alla definizione e sperimentazione di metodi formali per il trattamento dell'informazione musicale, sia a livello simbolico che sub-simbolico. Accanto alle attività di ricerca, principalmente sostenute dal Consiglio Nazionale delle Ricerche, si svolge annualmente un corso complementare di "Informatica Musicale" nell'ambito del secondo biennio del Corso di Laurea in Scienze dell'Informazione.

Il patrocinio di due entità della IEEE Computer Society, una internazionale e specifica del settore (la Task Force on Computer Generated Music) e una nazionale di interesse generale (la North Italy Section) incoraggia i ricercatori del campo informatico-musicale e testimonia l'ormai avvenuto riconoscimento internazionale della maturità disciplinare raggiunta.

I vari contributi presentati al Colloquio nelle varie forme di paper, poster, video, dimostrazioni e composizioni musicali sono tutti documentati negli Atti e organizzati per capitoli corrispondenti alle principali tematiche. Di notevole entità, come consueto, i capitoli sull'elaborazione numerica del segnale audio e sui sistemi per l'elaborazione musicale (workstation e strumenti software e hardware) e di attuale particolare interesse i capitoli sulla percezione e sui sistemi multimediali, realtà virtuale e spazializzazione.

Alla realizzazione del X Colloquio di Informatica Musicale hanno contribuito molti enti, società e soprattutto persone; desidero qui ringraziare vivamente: il Prof. Denis Baggi, chair della IEEE Computer Society Task Force on Computer Generated Music; il Prof. Roberto Negrini, chair della IEEE Computer Society North Italy Section; la Civica Scuola di Musica di Milano; il Comitato Scienze e Tecnologie dell'Informazione del Consiglio Nazionale delle Ricerche; la Direzione del Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" del Consiglio Nazionale delle Ricerche; la Silicon Graphics spa, nella persona del Dr. Pierpaolo Muzzolon; la Intersoft srl, nella persona del Dr. Rubens Malloggi; i membri dei Comitati Scientifico e Musicale del X Colloquio di Informatica Musicale; i membri del Comitato Organizzativo, M° Angelo Paccagnini, Dr.ssa Isabella Pighi e Dr. Dante Tanzi; i collaboratori e studenti del LIM - Laboratorio di Informatica Musicale che hanno dato una mano per gli aspetti organizzativi; il direttore, Prof. Giancarlo Mauri, e il personale non docente del Dipartimento di Scienze dell'Informazione; tutti gli autori dei contributi che, con il loro lavoro, hanno permesso di svolgere un'edizione del Colloquio così ricca di contenuti scientifici e musicali innovativi e di alto livello.

Milano, 20 ottobre 1993

PRESENTAZIONE

Lelio Camilleri

Presidente

AIMI - Associazione di Informatica Musicale Italiana

L'AIMI (Associazione di Informatica Musicale Italiana) compie il dodicesimo anno di vita e vede nello stesso anno l'organizzazione del X Colloquio di Informatica Musicale. Non è che i numeri abbiano molta importanza, ma la longevità e il continuo sviluppo delle attività coordinate dall'AIMI indica come questa associazione non è nata solo per riunire un gruppo di ricercatori e musicisti che lavorano in questo settore. Il suo obiettivo principale è quello di promuovere la crescita e lo sviluppo delle attività scientifiche e musicali nel campo dell'applicazione delle nuove tecnologie alla musica.

In questi anni l'AIMI ha organizzato, in prima persona o come co-organizzatore, workshop e convegni nazionali e internazionali. Ha, ovviamente, organizzato i Colloqui di Informatica Musicale il cui contenuto si è via via integrato con contributi di ricercatori e musicisti stranieri che hanno dato sempre più un carattere internazionale a queste manifestazioni. Infatti, delle 28 comunicazioni che compongono le sessioni di questo colloquio un terzo sono di ricercatori stranieri. E vi è una presenza di questi contributi anche nelle dimostrazioni, nei poster e nei concerti.

Questo è una prima realizzazione dell'auspicio di un'apertura dei nostri incontri ad un respiro europeo per poter conoscere e confrontarci con il lavoro svolto in altri paesi, che il precedente Presidente dell'AIMI, Giovanni De Poli, faceva nel IX Colloquio svoltosi a Genova. A De Poli devono andare i ringraziamenti dei soci per aver ben operato, fra tante cose, a intraprendere la giusta strada per il raggiungimento di questo obiettivo.

Per quanto riguarda il contenuto del convegno, esso si articola in una serie di sessioni che interessano sia argomenti prettamente tecnologici che musicali. Due di queste focalizzano l'attenzione su due sistemi di notevole interesse quali la Stazione di Lavoro Musicale Intelligente e la MARS. La

parte scientifica del colloquio viene completata da una serie di dimostrazioni e sessioni di poster.

Un altro aspetto di sicuro interesse è il Tutorial sugli "Standards in Computer Generated Music" seguito da un panel sponsorizzato dalla IEEE Computer Society Task Force on Computer Generated Music. La presenza come sponsor del convegno della Task Force, non è un caso dato che la sua recente nascita è dovuta anche alla collaborazione della comunità informatico/musicale Italiana.

Per quanto riguarda la parte musicale, il colloquio prevede l'esecuzione di 12 lavori che rappresentano distinti aspetti del rapporto fra tecnologia e composizione.

Per finire, vorrei ringraziare il comitato organizzatore, i comitati scientifico e musicale, gli sponsor, oltre che gli autori, per il loro prezioso contributo alla riuscita di questa manifestazione.

Firenze, 20 ottobre 1993

INDICE

Introduzione Goffredo Haus, Direttore Scientifico LIM-DSI	5
Presentazione Lelio Camilleri, Presidente AIMI	7
Indice	9
 Capitolo 1: TUTORIAL	
D. Sloan <i>From DARMS to SMDL, and Back Again</i>	19
 Capitolo 2: TEORIA MUSICALE, COMPOSIZIONE E MUSICOLOGIA	
M. Baroni, L. Finarelli <i>Alcune osservazioni sulla esecuzione della Quinta Sinfonia di Beethoven</i>	31
L. Camilleri, F. Carreras, F. Giomi <i>Sistemi esperti in musicologia: un prototipo per l'analisi TIME-SPAN reduction</i>	36
A. De Matteis, G. Haus <i>Formalizzazione di strutture generative all'interno de "La sagra della primavera"</i>	48
B. Fagarazzi, M. Sebastiani <i>Using Self-Affine Fractal Coding to Model Musical Sequences</i>	55
U. Merlone <i>Analisi statistiche nel riconoscimento degli intervalli</i>	59
S. Sargentì <i>Definizione di reti di Petri per l'analisi della musica elettroacustica</i>	63

N. Zahler <i>The Compositional Process and Technological Tools: an Appraisal of Algorithmic Composition as it Relates to Compositional Process</i>	67
Capitolo 3: PERCEZIONE	
D. R. Keane, L. L. Cuddy, C. A. Lunney, J. Dufton <i>The Perception of Musical Structure and Time</i>	79
M. Leman <i>Tone Center Attraction Dynamics: an Approach to Schema-Based Tone Center Recognition of Musical Signal</i>	86
Capitolo 4: RETI NEURALI	
G. U. Battel, R. Bresin, G. De Poli, A. Vidolin <i>Automatic Performance of Musical Scores by Means of Neural Networks: Evaluation with Listening Tests</i>	97
G. De Poli, P. Prandoni, P. Tonella <i>Timbre Clustering by Self-Organizing Neural Networks</i>	102
M. Johnson <i>Neural Networks and Style Analysis: a Neural Network that Recognizes Bach Chorale Style</i>	109
P. Toivainen <i>Modelling Harmony-Based Jazz Improvisation: an Artificial Neural Network Approach</i>	117
Capitolo 5: ELABORAZIONE NUMERICA DI SEGNALI	
M. Barutti, G. Bertini <i>Una nuova tecnica di sintesi additiva basata sulla trasformata inversa di Fourier</i>	127
L. Bazzanella, G. B. Debiasi <i>Analisi dell'effetto del tocco sul transitorio d'attacco dei suoni di un organo a canne a trasmissione meccanica</i>	134

A. Bernardi, G. P. Bugna, G. De Poli <i>Sound Analysis Methods Based on Chaos Theory</i>	142
A. Chandra <i>Counterwave. A Program for Controlling Degrees of Independence between Simultaneously Changing Waveforms</i>	151
A. Di Scipio, G. Tisato <i>Granular synthesis with Interactive Computer Music System</i>	159
S. Dubnov, N. Tishby, D. Cohen <i>Bispectrum of Musical Sounds: an Auditory Perspective</i>	166
C. Lippe <i>Real-time Control of Granular Sampling via Nonlinear Processes Using the IRCAM Signal Processing Workstation</i>	178
S. Mariuz <i>A Program for Analysis, Separation and Synthesis of Musical Signals Spectrum</i>	184
A. Pellecchia, A. de Vitis <i>Sintesi Polare: applicazioni in campo musicale di filtri digitali operanti al limite della stabilità</i>	188
A. Piccialli, S. Cavaliere, I. Ortosecco <i>Analysis, Synthesis and Modification of Pseudoperiodic Sound Signals by Means of Pitch Synchronous Techniques</i>	194
D. Rocchesso <i>Multiple Feedback Delay Networks for Sound Processing</i>	202
D. Rocchesso, F. Turra <i>A Real-Time Clarinet Model on MARS Workstation</i>	210
Z. Settel, C. Lippe <i>FFT-based Resynthesis for Real-Time Transformation of Timbre</i>	214

Capitolo 6: WORKSTATION MUSICALI

Sezione 6a: Stazione di Lavoro Musicale Intelligente

A. Camurri <i>The Cognitive Level of the Intelligent Music Workstation</i>	225
A. Camurri, A. Cartoncini, M. Frixione, C. Innocenti, A. Massari, R. Zaccaria <i>Toward a Cognitive Model for the Representation and Reasoning on Music and Multimedia Knowledge</i>	231
J. Chareyron, D. Rizzi <i>Due ambienti sperimentali dedicati alla sintesi LASy</i>	244
P. Fischetti <i>PC-Music: evoluzione del linguaggio CMUSIC per ambiente MS-DOS</i>	248
G. Haus, I. Pighi <i>"Stazione di Lavoro Musicale Intelligente": l'ambiente integrato Macintosh-NeXT</i>	254
G. Haus, A. Sametti <i>L'ambiente per l'analisi/re-sintesi di partiture della "Stazione di Lavoro Musicale Intelligente"</i>	262
C. Massucco, M. Mercurio, G. Palmieri <i>Real-Time Processing and Performance Using WinProcne/HARP</i>	270

Sezione 6b: MARS

P. Andrenacci, F. Armani, A. Prestigiacomo, C. Rosati <i>APPLI20: a Development Tool for Building MARS Application with an Easy to Use Graphical Interface</i>	277
E. Favreau, S. Sapir <i>La stazione MARS: dalla progettazione di algoritmi alla realizzazione di ambienti esecutivi dedicati</i>	285
E. Guarino, R. Bessegato, E. Maggi <i>Celle-funzione per la realizzazione di sistemi musicali elettronici</i>	293

E. Maggi, A. Prestigiacomo <i>Portability of the MARS System</i>	301
---	-----

Sezione 6c: Altre workstation

G. Bertini, D. Fabbri, M. Marani, L. Tarabella <i>MUST C 25 - Stazione di lavoro musicale con schede DSP</i> <i>Leonard C25</i>	307
---	-----

P. Prévot, A. Debayeux <i>Constraint Satisfaction Programming in Computer Aided Composition on a Highly Gestual Devoted System, Based on a VME- Multi-Processor Joining True UNIX and Real-Time</i>	311
--	-----

Capitolo 7: STRUMENTI S/W E H/W PER LA COMPOSIZIONE E LA PERFORMANCE

S. Bettini <i>Music 5 Mac</i>	321
----------------------------------	-----

R. Bresin <i>MELODIA: a Program for Performance Rules Testing, Teaching, and Piano Scores Performing</i>	325
---	-----

N. Larosa, C. Rosati <i>MEDUSA: a Powerful MIDI Processor</i>	328
--	-----

M. Laurson <i>PWConstraints</i>	332
------------------------------------	-----

A. Provaglio <i>SoundLib 2.0 - Una libreria di classi C++ per l'elaborazione di segnali audio campionati</i>	336
---	-----

G. Ramello <i>"HIPPOPOTAMUS": un sistema di performance interattivo</i>	340
--	-----

L. Tarabella, G. Bertini, M. Romboli <i>Le Twin Towers: un dispositivo per esecuzioni interattive di computer music</i>	344
--	-----

Capitolo 8: SISTEMI MULTIMEDIALI, REALTA' VIRTUALE, SPAZIALIZZAZIONE

A. Belladonna, A. Vidolin <i>Applicazione MAX per la simulazione di sorgenti sonore in movimento con dispositivi commerciali a basso costo</i>	351
A. Camurri, F. Giuffrida, G. Vercelli, R. Zaccaria <i>A System for Real-Time Control of Human Models on Stage</i>	359
S. T. Pope and L. E. Fahlén <i>The Use of 3-D Audio in a Synthetic Environment: an Aural Renderer for a Distributed Virtual Reality System</i>	366

Capitolo 9: STUDIO REPORT

P. Dutilleux <i>Center for Art Mediatechnology Karlsruhe: the Institute for Music and Acoustics</i>	379
L. Gamberin, S. Mosca <i>La biblioteca, il computer e la musica</i>	382
L. Garau, G. Tedde <i>L'attività dell'Associazione Ricercare ed il suo studio per la ricerca musicale e artistica</i>	387

Capitolo 10: COMPOSIZIONI MUSICALI

Ludger Bruemmer <i>The Effect of Digital Synthesis Language on the Conception and Process of Composition</i>	393
Luigi Ceccarelli <i>DOPPIO SOLO</i>	397
Fabio Cifariello Ciardi <i>FINZIONI</i>	401
Alessandro Cipriani <i>VISIBILI</i>	404

James Dashow	
<i>RECONSTRUCTIONS</i>	407
A. Di Scipio	
<i>Sulla composizione di ZEITWERK (l'orizzonte delle cose)</i>	410
Amedeo Gaggiolo, Silvia Dini	
<i>ANIMALI IN SOFFITTA</i>	414
Francesco Galante	
<i>METAFONIE</i>	418
Francesco Giomi	
<i>Alcune riflessioni intorno al brano elettroacustico</i>	
<i>CHROMATISM</i>	422
David Keane	
<i>WERVELWIND</i>	425
Cort Lippe	
<i>A Composition for Clarinet and Real-Time Signal Processing: Using Max on the IRCAM Signal Processing Workstation</i>	428
Matteo Pennese	
<i>IHADA</i>	433

Capitolo 1

TUTORIAL

From DARMS to SMDL, and Back Again

Donald Sloan

Music Department
Ashland University
Ashland, OH 44805 (USA)
fax 419-289-5333
email authreen@class.org

Abstract.

The information standards HyTime and SMDL (ISO/IEC 10744 and ISO/IEC CD 10743, respectively) were created to handle representation and manipulation of hypermedia documents. HyTime was designed to handle any combination of media, while SMDL is being designed as an application of HyTime specific to music information. In part, SMDL is intended to help those working with databases of musical information. This paper contains a demonstration of how SMDL can provide a standard for the exchange of database information, regardless of the original database format. The information captured in one such music representation scheme, DARMS, can be shown to be able to map into SMDL, and vice versa. In a like manner, other representation schemes, including but not limited to SCORE, MUSTRAN, ENIGMA, MIDI and CCARH can have a standard base for information exchange,

without necessitating a separate pair of translators or interpreters be designed for each pairwise combination of schemes. Instead, each format would only need two such translators: one from the original format into SMDL, and one from SMDL into the original format. This would greatly reduce the number of translators necessary today, and would ensure that in the future, should new representation schemes appear, the need for translators would only grow arithmetically, not geometrically.

Introduction.

Since 1986, the X3V1.8M committee of the American National Standards Institute (ANSI) has been undergoing a public standards process for the representation of static and dynamic information in hypertext and multimedia documents. This process has produced two related standards. Hypermedia/Time-based Structuring Language (HyTime) has already been

approved by the International Standards Organization (ISO), of which ANSI is the United States representative. HyTime has been assigned the number ISO/IEC 10744:1992. Standard Music Description Language (SMDL) is still in development, and in its current draft form bears the number ISO/IEC CD 10743.

Both of these standards rely on an existing standard, SGML (ISO/IEC 8879:1986), a markup language which allows documents to be tagged in such a way as to allow representation of virtually any document architecture, by the representation of semantic and syntactic information. SGML is used today by many publishers world-wide, both in the public and private sectors. HyTime is an application of SGML; that is, HyTime relies on certain functions and capabilities outlined in the SGML standard. A HyTime user could design a Document Type Definition (DTD) based on SGML conventions, plus those parts of HyTime that are needed for that type of document. Many different instances of a given document type can use the same DTD. In that sense, HyTime is an example of an object-oriented approach; the objects (called elements in SGML) are defined in the DTD, while their actual attributes can vary from instance to instance. In a similar manner, SMDL is an application of HyTime. It uses a DTD that relies on HyTime's

facilities, plus those unique to SMDL.

It is important to note that both HyTime and SMDL are enabling standards. They place no restrictions on the content of what is represented, nor do they enforce a single kind of document architecture. By following the standards, one may gain those things that a standard should confer: a representation scheme that is public, a means for expressing the structure of information, and a set of tools for performing useful operations on the information, such as data queries and hypertext links. The standards do not constrain document styles; for example, two different publishers may use different formats for chapter headings. The tagging of a certain datum as a chapter heading is, however, an important piece of information, regardless of what a user may choose to do with it. This principle guided the developers of HyTime and SMDL; a proper level of abstraction and an appropriate set of tools will allow any author to retain both the data content and the document structure in a useful way. These standards do not force an author to use all of the facilities of the standards. One can include a non-SGML document, and label it as such; one would lose much of the power of the standards, but at least the choice is up to the user.

HyTime and SMDL are intended for use by virtually the

entire community of information processing, from paper to electronic publishers, from hypertext documents to multimedia documents, from authors to end users. It is hoped that developers will create a set of software tools similar to what now exists for SGML, so that the intricacies of HyTime and SMDL will be hidden behind an interface that allows a wide range of users to transparently author and read HyTime and SMDL documents. The HyTime standard can be obtained from national standards organizations, as well as from the ISO/IEC Copyright Office, Case Postale 56, CH-1211 Génève 20, Switzerland. Annex B of the HyTime standard lists additional sources of information. SMDL is still in development, and as such, is in draft form.

Why Two Standards?

Although this project began with only one standard in mind, it became clear that the requirements of the music community were far more detailed than those of the hypermedia community in general. HyTime contains a highly abstract approach to information architecture, consisting of several modules representing these functions: 1) a way to measure the position and extent of objects, 2) a way to address objects that cannot be

given unique identifiers in SGML, 3) a way to link objects, whether or not in the same document, 4) a way to schedule events so that not only the proper sequence, but also the proper relationship between events is maintained, and 5) a way to render a particular instance of such a schedule. The scheduling and rendering modules owe a particular debt to the music world; it was determined that the scheduling model for musical events was at least as sophisticated as that for any other medium, and thus any model that could represent musical events was robust enough for virtually any schedule. The content of SMDL, therefore, contains only those items unique to music representation. The time model is already present from SMDL's inheritance of HyTime facilities, meaning that an SMDL document will invoke HyTime elements to represent durations. Those HyTime modules that are not applicable to music representation may be left out; there are different levels of conformance depending on user needs. Thus, if a user does not need hypertext linking facilities, this need not be included. This feature of modular architecture will make it easier for those using HyTime and SMDL for music applications.

HyTime's Time Model.

HyTime uses its scheduling module to represent sequences of events, durations, and other temporal relationships between objects. In HyTime, a work may be represented by a number of coordinate axes, with each axis representing a temporal or spatial dimension. Each event has a position and extent on an axis. For temporal events, this corresponds to start time and duration. Thus, it may be possible to show the staging of an opera with a temporal axis showing the flow of the music, and two spatial axes showing where each character is on stage at any given moment of the music.

These axes together make up what is called a finite coordinate space, or FCS. The measurement system for each axis is defined by the DTD, and remains fixed for that work. An axis may use a

standard measurement unit (SMU) such as meters, seconds, etc., or a virtual measurement unit, with its temporal or spatial meaning defined by the user. The latter system is what would be useful to represent music that is written in Common Western Notation. When musicians think of half notes and quarter notes, these do not have an absolute duration in seconds until the time of performance; even then, no two performances would be the same temporally. Thus, a logical representation in virtual units is what is required to properly capture the musical information.

While it is beyond the scope of this paper to give a tutorial extensive enough to teach fluency in HyTime scheduling, a certain amount of this can be shown in the following way: a potential DTD for an SMDL document could contain the following information:

```
<!DOCTYPE work SYSTEM "smdl.dtd" [
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32 >
<?HyTime MODULE base
    desctxtx dvlist lextype refct1 >
<?HyTime MODULE measure
    axismdu dimref fcsmdu HyFunk HyOp markfun >
<?HyTime MODULE sched
    grpdex >
<?HyTime MODULE rend
    modify patch profun project >
<!NOTATION virtime PUBLIC -- virtual time --
    "+//ISO/IEC 10744//NOTATION Virtual Time Unit//EN"
>
<!ENTITY tactvtu -- number of vtus per tactus (beat) --
"80640" >

<!ENTITY % av.wxdm -- dimension of wfaxis -- "4294967295">
```

```

<!ENTITY % av.wxfm  -- mdu def for workfcs -- "'SIsecond 1
1000'" >
<!ENTITY % av.wxbg  -- basegran of wfaxis -- "msec" >
<!ENTITY % av.wxgh  -- gran2hmu of wfaxis -- "'1 1'" >
<!ENTITY % av.wxpg  -- pls2gran of wfaxis -- "'1 1'" >

<!ENTITY % av.cxdm  -- dimension of mustime axis --
"4294967295">
<!ENTITY % av.cxbg  -- basegran of mustime axis; -- "vtu"
>
<!ENTITY % av.cxgh  -- gran2hmu of mustime axis -- "'1 1'"
>
<!ENTITY % av.cxpg  -- pls2gran of mustime axis -- "'1
&tactvtu;'" >
<!ENTITY % av.cxfm  -- mdu def for mustime axis --
"'virtime 1 1'" >

```

The first line of code gives a name to the DTD. The next group of lines names those HyTime modules, and within each module, those functions which will be supported by this DTD. The notation is declared as virtual time, and the units of measurement virtual time units, or vtus. Each beat or tactus is given a total of 80640 vtus, which ensures that equal division of the beat into 2, 3, 4, 5, 6, 7, 8, 9, etc. units can be achieved with integer totals in each case. This number may change according to user needs; for example, there are certain Chopin works in which a beat is divided into 17 notes. In such a case, the number of vtus per tactus could be a multiple of 17 as well. This would guarantee an integer number of vtus for each note, avoiding the problems inherent in floating-point calculations. One may always arrive at a safe ‘tactvtu’ number by taking a common multiple of the various divisions of the beat in a given

piece. The least common multiple would be most economical, of course, but since this DTD may serve several different pieces, it is probably safer to choose a larger multiple that will not have to be adjusted from piece to piece.

The next five entities define a real time axis, upon which a performance of the musical work could be represented. Notice that the measurements are in seconds and milliseconds, real time units. The last five entities define a virtual time axis, upon which the temporal information inherent in the score would reside. This axis is defined in terms of virtual time units, or vtus. The user may declare in another part of the document what the tactus represents. In this way, a single DTD may be flexible enough to handle several different types of pieces, or even a single piece in which the tactus changes.

These lines of code could be used potentially as a typical SMDL DTD, thus obviating the

need to redefine all of the HyTime terms for each work of music represented. There are other parts to this typical DTD that refer to resource tables and other SMDL devices; these have been omitted in this example in order to isolate those things relating directly to the time model, but would ordinarily be included in a typical SMDL DTD.

Since this DTD allows for an axis in virtual time, to capture the logical information, and an axis in real time, to capture a potential performance, there must be a method for mapping from one to the other. HyTime has elements in the rendition module called batons and wands to create this conversion. A baton is a schedule of projectors which can map virtual units to real units, with extensive facilities for determining how to reconcile one to the other. A wand is a schedule of modifiers which act upon HyTime objects. An example of a modifier would be a filter, either sonic or visual. The modifying abilities of these scopes could be defined by the user; for example, those interested in representing electronic music may find it useful to define scopes in terms of the formulas used to manipulate waveforms, rather than have to name a specific pitch for a sound that is being processed in such a way as to make pitch difficult to define. HyTime does not provide a standard set of tools to describe

sound manipulation; rather, it has a place in the standard for those who wish to describe their sound manipulation methods in a representation of their choosing.

From DARMS to SMDL...

The Digital Alternate Representation of Musical Scores, or DARMS, is one of several data content notations currently in use in the music world for the representation of music information. There exist musical databases encoded in DARMS, as well as other data content notations, such as SCORE, MUSTRAN, the CCARH code, MIDI and others. The inclusion of MIDI in this list is an illustration that not all musical information resides in scores; there is a need for capturing potential performances as well. Since humans use a musical score as a blueprint for performances, some people have believed that a data content notation that represents a score represents a performance as well. Any programmer can tell you that there is a good deal of interpretation that a machine must accomplish in order to render a score in actual sound. DARMS is one of those codes that captures the appearance of the musical symbols on a score, but requires an intelligence to give sonic definition to these symbols.

SMDL is being designed to cover both the visual aspects of music representation, such as scores, and the auditory aspects of music representation, such as gestural information for performances. There exist several domains for the information in an SMDL work. The set of logical information, that is, the temporal relationships, pitch information, and the like, is represented in a domain called the cantus. The purely visual phenomena such as how the musical information appears in a score, is in the visual domain. The purely audible phenomena, such as how the music is manifest in a certain performance, is in the gestural domain. Finally, manipulations that rely on the work's musical information, but are not properly part of the piece of music reside in an analytic domain. It is the cantus that contains that which uniquely defines the work; one could imagine taking the information in the cantus and creating several different-looking scores (e.g., two representations of Hebrew cantillation, one using the original tropes, and another in Western notation), or two different performances. And yet, in spite of the different ways of representing it, there remains only a single work of music in the cantus. This cantus represents the abstract information of a piece of music.

Thus, a DARMS-encoded score could be represented in

SMDL both in the cantus, which would contain pitch names, articulations and relative note proportions, and in the visual domain, which would contain such purely visual phenomena as clefs, articulation symbols, beaming information, etc. For each symbol in DARMS (or any other existing coding scheme, for that matter), there will be a corresponding way to capture all of the musical information in SMDL.

The latter point is crucial if SMDL is to be useful as a standard. SMDL should be able to represent all that we regard as musical information. In certain cases, this representation may be nothing more than an ability to specify the extent of a chunk of data, and a label for what it is. For example, rather than convert or translate a DARMS database into SMDL data, one can include it in an SMDL document as an uninterpreted section of data called "DARMS code." This will not allow many operations that SMDL could otherwise perform on musical data, but at least would provide a standard way to exchange data with proper clues as to what the data content notation is. Similarly, a bitstream of sound samples may remain undifferentiated in an SMDL document, but bear the label "digital sound samples," so that a device that could potentially use this information will know what it is.

...And Back Again.

Since SMDL can represent at least as much information as would be in a DARMS-encoded score, conversion from SMDL to DARMS would be a matter of using only that information which could be captured by DARMS. If one used only the information present in the cantus of an SMDL work, then there is no guarantee that the resulting DARMS translation would be exactly like the original that was translated to SMDL in the first place. This is because the cantus contains information about the exact pitch, but not how it is represented on a page. Thus, in order to get from an SMDL work back to the DARMS code with which one started, one would have to use both the cantus and a visual domain together. Fortunately, SMDL has an element in the visual domain in which any datum in that domain can be referenced to its corresponding logical information in the cantus. Thus, one can be sure that an event in the cantus and an event in the visual domain refer to the same musical object, and can be used together to translate to the corresponding DARMS code. The link direction of such a reference is from the visual domain to the cantus, not the other way around, since there are potentially limitless ways to represent the logical information

visually, but each visual symbol should refer uniquely to its logical information.

Why is it important to be able to translate from DARMS to SMDL, and vice versa? If one wanted to be able to share databases of musical information, without a standard, it would be necessary to create a pair of translators for each pairwise combination of codes. Thus, if there are ten codes commonly in use, a full translating capability would require 90 different translators. With a standard, one would only need a translator from each code to the standard, and back again. The same ten codes could be fully shared with only 20 different translators. As representation standards proliferate, the need for translators would grow geometrically without a standard, but only arithmetically with a standard. Such a standard need not crowd out all other methods of representation; rather, it can be used only to the extent that it is valuable to do so. It has been suggested that using an already-existing code would save the trouble of developing a standard such as SMDL. Yet, it has not been demonstrated that any existing code can capture all of the information necessary for both musical score and musical performance.

Current Status.

Unfortunately, development of SMDL has been slowed while its parent application, HyTime, undergoes the inevitable ironing out of wrinkles during its initial implementations. At the present time, SMDL is incomplete, and is not ready for developers to use. This will change shortly, as HyTime matures. In the meantime, those who have been using another coding system for databases can continue to use it; when SMDL is ready, a translator can convert the data. While SMDL will not include such a translator in the standard, for those codes where such a need is evident in the music community, it is hoped and anticipated that developers will create such tools as translators. The success of a music standard such as SMDL and the needs of the music community are thus intertwined.

Capitolo 2

TEORIA MUSICALE, COMPOSIZIONE E MUSICOLOGIA

ALCUNE OSSERVAZIONI SULLA ESECUZIONE DELLA QUINTA SINFONIA DI BEETHOVEN

Mario Baroni, Luigi Finarelli

Dipartimento di Musica e Spettacolo, Università di Bologna
via Galliera,3
I-40125 Bologna (Italy)
fax +39 51 231183
E-mail g3ubouc1@icinca

Abstract

This paper presents a tentative approach to the study of an orchestral musical performance. The opening theme of Beethoven's Fifth Symphony has been analyzed. Timing only is taken into consideration in this first phase of the analysis. Some results are presented, but mainly methodological problems are discussed.

a) Obiettivi

Lo studio della letteratura sulla esecuzione musicale dimostra che, dopo il periodo pionieristico degli anni scorsi, la riflessione scientifica sta raggiungendo oggi una maggiore sicurezza e stabilità. La consapevolezza di questo stato di cose ci ha suggerito il tentativo di saggiare, a titolo sperimentale, un terreno che, per almeno due aspetti, non è stato finora frequentemente esplorato.

Il primo aspetto riguarda l'impostazione stessa del problema. Di norma la letteratura sull'esecuzione musicale ha il proprio retroterra nella psicologia o nella psicoacustica e il suo scopo primario è quello di esaminare in dettaglio il rapporto tra i fenomeni sonori e la loro elaborazione cognitiva. Nel nostro caso l'obiettivo è invece quello di cominciare a vedere in che misura i risultati finora acquisiti possano essere utilizzabili per rispondere a domande provenienti dal campo della musicologia.

Il secondo aspetto riguarda le fonti sonore esaminate. Nella grande maggioranza dei casi presi in considerazione in letteratura, la coscienza dei limiti dell'odierna tecnologia ha opportunamente suggerito di attenersi all'esame di frammenti eseguiti su strumenti monofonici o sul pianoforte. Anche nel nostro caso ci siamo attenuti ad esempi monofonici, ma abbiamo

reso in considerazione uno strumento complesso come l'orchestra.

In questo primo approccio ci siamo limitati ad alcune indagini semplici, anche perché l'intenzione iniziale è quella di toccare con mano fino a che misura sia oggi possibile addentrarsi in un campo di questo genere. Abbiamo preso in considerazione uno dei temi più famosi di tutta la musica occidentale: l'incipit della Quinta Sinfonia di Beethoven eseguito da Solti, Abbado, Karajan, Toscanini, Knappertsbusch e Harmoncourt.

b) Strumenti tecnologici, misure, problemi di metodo

Per raccogliere i dati descritti ci siamo serviti delle funzionalità di editing digitale del sistema Sonic Solutions [1] che, con l'uso di 12 tracce parallele virtuali, consente anche una immediata comparazione visiva tra i profili d'ampiezza dei campionamenti.

Nell'analizzare le esecuzioni di musica orchestrale abbiamo incontrato una serie di problemi di non facile soluzione: la tecnica e l'epoca della registrazione, la risposta acustica dell'ambiente e inoltre i tempi di risposta tipici dell'orchestra dovuti alla vastità dell'insieme, alle asincronie degli attacchi, ai transitori d'attacco e al tempo di trasmissione del segnale acustico nella sala.

Abbiamo però considerato che le singole registrazioni sono state riascoltate e licenziate dal direttore, in quanto interprete principale, come esecuzioni ben definite sia dal punto di vista artistico che storico.

Per il momento ci siamo limitati solamente a misure di tempo, ma contiamo di approfondire l'indagine prendendo in considerazione anche i fattori ampiezza e timbro.

Tale limitazione riduce fortemente anche le possibilità di interpretazione dei risultati. Il nostro obiettivo musicologico è infatti quello di ricostruire, interpretando i dati fonici, gli orientamenti espressivi che ciascun direttore possiede nel momento in cui utilizza i margini di libertà che il testo scritto gli concede. La prima condizione per iniziare un lavoro di questo tipo è che per lo stesso frammento si possano eseguire rilevamenti su più parametri. Esistono infatti correlazioni documentabili fra scelte di durata di vari tipi e scelte d'ampiezza (e probabilmente anche scelte timbriche) che obbediscono a principi di coerenza espressiva [2]. La limitazione a un solo parametro (quello delle durate) riduce i margini di verifica della nostra interpretazione.

L'unica possibilità di indagare sulla coerenza delle scelte è data nel nostro caso dal confronto delle durate in

frammenti diversi della stessa esecuzione o dal confronto delle durate dello stesso frammento in esecuzioni diverse. Ma è chiaro che in questi casi esistono interferenze di variabili esterne non controllate che rendono l'interpretazione malsicura. Tuttavia, come abbiamo detto, il nostro scopo è solo quello di cominciare a esplorare il campo. In linea di massima ci limiteremo dunque a segnalare problemi più che a dare risultati.

c) Confronto fra il tema e le sue ripetizioni

Nella partitura il tema principale (batt. 1-5) si ripresenta esattamente uguale in altre due occasioni, la ripetizione dopo il ritornello di batt. 124 e la ripresa a batt. 248.

Il confronto fra questi tre frammenti solleva un problema ben noto agli studiosi del campo: l'esecuzione non si limita a riprodurre le note scritte, ma si applica sempre a una struttura. Prima di venire eseguito ogni frammento deve essere interiormente pensato e definito dal suo esecutore mediante l'assegnazione di un preciso ruolo funzionale nella sintassi del brano, nonché di particolari caratteristiche espressive. Solo a questo punto l'esecutore decide le proprie scelte interpretative [3].

Ora, le tre comparsate del tema nel primo movimento hanno tre funzioni sintattiche ben

diverse: la prima ha il ruolo dell'enunciazione iniziale, la seconda è un ritornello e la terza (ripresa) arriva solo al culmine della frase che la precede. L'ultima comparsa ha dunque due funzioni: è ripresa rispetto alla macroforma e climax rispetto alla frase in cui è inserita. Le durate dei tre esempi variano come indicato in Fig. 1.

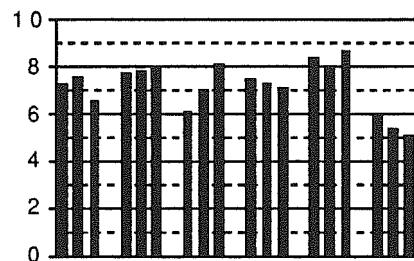


Fig. 1. Durata in secondi del tema del primo movimento della V sinfonia di Beethoven nelle sue ripetizioni; da sinistra: Solti, Abbado, Karajan, Toscanini, Knappertsbusch, Harnoncourt.

Karajan, Abbado e Knappertsbusch, come si vede, rallentano il tempo nella terza ripetizione. E' assai probabile che questo rallentamento possa essere interpretato in funzione del climax, poiché in moltissimi esempi la sottolineatura enfatica presuppone appunto un rallentamento [4]. Ciò andrebbe però confermato da altri rilevamenti sull'ampiezza e sul timbro. Non c'è poi da stupirsi che Solti, Toscanini e Harnoncourt adottino invece una strategia esattamente opposta. Sundberg [5] segnala a questo proposito casi significativi che

chiama "sinonimie" (stesso effetto ottenuto con strutture diverse o anche opposte). Anche in questo caso tuttavia resterebbe da decidere, ed è possibile farlo sulla base della teoria "movimento-emozione" enunciata da Gabrielsson [6], che tipo di proprietà emozionali attribuire a questa accelerazione. Ma per far ciò occorrerà utilizzare altri più precisi dati.

d) Confronto fra le diverse esecuzioni del tema

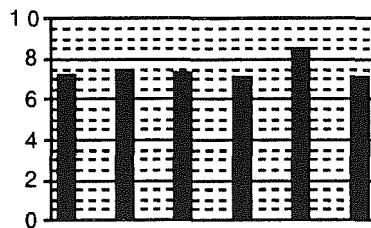


Fig. 2. Durata totale in minuti del primo movimento della V sinfonia di Beethoven; da sinistra: Solti, Abbado, Karajan, Toscanini, Knappertsbusch, Harnoncourt.

Il tema è stato suddiviso in due elementi (sol-sol-sol-mi, fa-fa-fa-re) di ciascuno dei quali abbiamo verificato le durate.

In questo contesto ci interesseremo soprattutto dell'enunciazione del tema (tralasciando le durate di ritornello e ripresa che pure sono indicate in figura). Al contrario di quanto ci si potrebbe aspettare l'enunciazione più veloce non è affatto quella di Toscanini: le esecuzioni di Karajan e Harnoncourt sono molto più brevi.

Knappertsbusch conserva invece la sua fama di esecutore solennemente lento. Se tuttavia si confrontano con questo dato le durate globali dell'esecuzione (Fig. 2) si vede chiaramente che la scelta del tempo per il tema non coincide con quella della cosiddetta pulsazione media [6]: qui Toscanini riottiene il suo primato e Karajan si dimostra notevolmente più lento di lui. Dunque la scelta del tempo indicata in Fig. 3 non si riferiva alla sinfonia in quanto tale, ma proprio al suo tema.

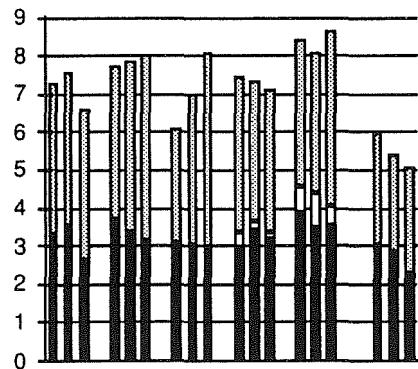


Fig. 3. Durate in secondi dei due elementi del tema del primo movimento della V sinfonia di Beethoven (sol-sol-sol-mi in nero, pausa in bianco e fa-fa-fa-re in grigio); da sinistra: Solti, Abbado, Karajan, Toscanini, Knappertsbusch, Harnoncourt

E' ovvio anche qui che la pura e semplice durata non può dare riferimenti per quanto riguarda le intenzioni interpretative. E' assai probabile ad esempio che fra la "velocità" di Harnoncourt e quella di Karajan possa esistere quel rapporto di opposizione che

Clarke [2] definisce come “understatement” (eccesso di ritegno) vs. “emphasis”(grande impeto) , ma per dire questo, ripetiamo, occorrono altri indizi fonici che ancora non abbiamo raccolto.

Per quanto riguarda il rapporto fra primo e secondo elemento del tema (Fig. 3) il sol-sol-sol-mi sembra in genere decisamente più veloce (teso? impetuoso?) del fa-fa-fa-re. Ma tale interpretazione andrà precisata. Le manca infatti un dato importante e cioè la durata del mi e del re coronati rispetto alle tre note ribattute. Se, come unanimemente ribadiscono tutti gli studiosi del settore, la specificità dell'esecuzione consiste nelle deviazioni rispetto alla durata grafica, che va intesa come una *categoria* di durata e non come una durata reale [7], allora la corona è già un segno squisitamente interpretativo che Beethoven stesso suggerisce ai suoi esecutori. Ma c'è chi lo prende più e chi meno drammaticamente. Purtroppo però abbiamo avuto dubbi sulla misura di questa specifica nota, perché il suo attacco e soprattutto la sua estinzione pongono seri problemi interpretativi. La pausa di Toscanini e Knappertsbusch è evidentemente dovuta alla tecnica di registrazione. Ma per gli altri esecutori quanto durava esattamente?

Bibliografia

- [1] The Sonic System, *User Manual*, Software version 2.0.5, Sonic Solutions, San Rafael, 1993
- [2] E.F.Clarke: “*Generative principles in music performance*”, in “*Generative Processes in Music*”, J.A.Sloboda ed., Clarendon Press, Oxford, pp.1-26, 1988
- [3] B.H.Repp: “*Patterns of expressive timing in performances of Beethoven minuet by nineteen famous pianists*”, Journal of Acoustic Society of America, Vol.88, N.2, pp. 622-641, 1990
- [4] J. Sundberg: “*Music performance research: An overview*”, in “*Music, Language, Speech and Brain*”, J. Sundberg, L.Nord, R.Carlson eds., Macmillan, London, pp. 173-183, 1991
- [5] J.Sundberg: “*Computer synthesys of music performance*”, in “*Generative Processes in Music*”, J.A.Sloboda ed., Oxford,Clarendon Press, pp. 52-69, 1988
- [6] A. Gabrielsson: “*Timing in music performance and its relations to music experience*”, in “*Generative Processes in Music*”, J.A.Sloboda ed., Clarendon Press, Oxford, pp.27-51, 1988
- [7] E.F.Clarke: “*Categorial Rhytm perception: an ecological perspective*”, in “*Action and Perception in Rhytm and Music*” A.Gabrielsson ed., Stockholm, Royal Swedish Academy of Music, pp. 19-34, 1987

SISTEMI ESPERTI IN MUSICOLOGIA: UN PROTOTIPO PER L'ANALISI TIME-SPAN REDUCTION

L. Camilleri, F. Carreras, F. Giomi

Divisione Musicologica CNUCE/C.N.R.
Conservatorio di Musica "L. Cherubini"
P.zza Belle Arti, 2
I-50122 Firenze (Italy)
Tel.: +39-55-282105
E-mail: conserva@vm.idg.fi.cnr.it

Abstract

The project of developing of an expert system for tonal harmonic analysis has started from some theoretical considerations about the existent theories (mainly Lerdahl and Jackendoff's GTTM) and tried to attain two purposes: 1) the creation of an analysis environment, to be easily improved and amplified; 2) the testing and improvement of theory contents, together with the verification of the integration among different theoretical assertions; for example, the theory of tonal pitch spaces, proposed again by Lerdahl in 1988 can give us useful hints in finding the harmonic modulation points. The system is composed of an harmonic lexicon and a rule system. At first the system divides the piece into time slices, as in the analytical methodology; they are catalogued according to their harmonic properties and at

this stage the ambiguous or unconventional slices (with regard to their relation to a triadic chord) are interpreted as probable chords and labelled. Then, the probable cadential points are found out: this part of the system deals with the representation of the structural properties of the "harmonic phrase". After this surface analysis, the system starts to abstract the distinct, hierarchical levels up to the deepest one using different rules and also different degrees of influence for each rule at each level. The implementation makes use of different AI techniques based on rules: the algorithmic part is written in Pascal while the facilities of the IBM shell ESE are particularly useful for speeding up the development process. Forward and backward chaining as well as rule execution control statements are used.

Introduzione

La ricerca è stata sviluppata dalla Divisione Musicologica del CNUCE/C.N.R. presso il Conservatorio di Musica "L. Cherubini" di Firenze avendo come obiettivo l'applicazione di tecniche di intelligenza artificiale all'analisi armonica di musica tonale occidentale.

Il lavoro si è articolato in una prima fase di approfondimento, ampliamento e integrazione di alcune recenti teorie musicali al fine di creare un ambiente per l'analisi armonica; è stato successivamente sviluppato un prototipo di sistema esperto che affonda le sue basi teoriche principalmente nella teoria generativa della musica (GTTM) di Fred Lerdahl e Ray Jackendoff [1][2]. Obiettivo finale è l'espletamento automatizzato di uno dei paradigmi della teoria, la riduzione dell'intervallo temporale (*time span reduction*): si tratta di una serie di regole e di criteri che permettono la progressiva riduzione di un brano, nell'ambito degli intervalli di tempo di volta in volta individuati, nei suoi elementi armonicamente più importanti. Questo procedimento di riduzione, articolato su diversi livelli, consente un processo di analisi strutturale i cui assunti teorici sono stati da noi ampliati e confrontati con i principi derivati da altre teorie musicali ed integrati con alcune parti teoriche appositamente studiate.

Alla luce di queste considerazioni, con la progettazione e la

realizzazione del sistema ci siamo posti due obiettivi principali:

1) la creazione di un ambiente per l'analisi armonica, aperto a miglioramenti ed ampliamenti;
2) la verifica e l'arricchimento sia contenuti teorici impiegati sulla base dei risultati analitici derivati dall'utilizzo integrato di diverse fonti teoriche. Quello che può essere considerato il predecessore del sistema, sia per affinità negli strumenti di sviluppo impiegati che per il tipo di teoria musicale su cui si fonda, è il prototipo per l'individuazione della struttura a gruppi di brani melodici tonali realizzato sempre dal nostro gruppo di ricerca [3]. Alla base delle regole di segmentazione inserite troviamo la parte della teoria di Lerdahl e Jackendoff che si orienta sull'analisi di un fattore cruciale della comprensione musicale: la segmentazione, a diversi livelli, di una linea melodica in gerarchie di gruppi [1][2]; questa parte è stata integrata anche con altri apporti derivati da altre teorie di origine semiologica e psicologica. Mentre in ingresso si fornisce la codifica di una melodia tonale, l'uscita del programma è rappresentata dalla struttura gerarchica della suddivisione in gruppi di ogni livello, insieme agli elementi (sequenze di note o sequenze di gruppi di livello più basso) contenuti in ognuno di questi.

Non potevano non essere presi in considerazione ai fini della pro-

gettazione altri due importanti lavori in questo campo: i sistemi esperti di John Maxwell e di Kemal Ebcioğlu. Nel primo caso [4] si tratta di un prototipo, basato su regole, che esegue l'analisi della funzione armonica degli accordi di un brano di musica barocca mentre il sistema di Ebcioğlu [5] si basa su un processo di sintesi, anziché di analisi, per indagare il repertorio dei corali bachiani, armonizzati ed analizzati automaticamente con tecniche analitiche di impostazione per lo più schenkeriana.

Approcci analitici

La GTTM può rappresentare un modello di analisi che ben si presta per la sua natura ad essere implementato e di conseguenza ad essere sottoposto a verifiche ed eventualmente ad integrazioni ed ampliamenti [6]. Come primo aspetto, essa tenta di produrre delle descrizioni formali dei pezzi di musica analizzati. I risultati analitici possono essere usati come termine di confronto, in relazione alle comuni intuizioni musicali, per la verifica dei principi teorici asseriti. In secondo luogo, la natura della teoria è di matrice psicologica, dato che tenta di spiegare capacità di tipo cognitivo: l'attenzione è rivolta all'organizzazione strutturale che l'ascoltatore esperto attribuisce inconsciamente a un pezzo musicale e ai principi cognitivi "universali" attraverso i quali essa viene determinata.

Fra gli studiosi di musicologia computazionale che hanno tentato realizzazioni dirette degli spunti teorici della GTTM vale la pena di ricordare Michael Baker, noto per la costruzione di sistemi per l'analisi automatica basata sui raggruppamenti e per forme di interazione fra *grouping analysis*, parallelismo musicale e livelli di riduzione [7]; l'équipe del Brooklyn College con il sistema di regole e un modello connessionista per la scoperta delle strutture metriche e di raggruppamento di brani tonali [8]; recentemente anche Frode Holm ipotizza l'applicazione dei processi comunicanti sequenziali alla riduzione dell'intervallo temporale, proponendo un algoritmo per la riproduzione udibile dei livelli di analisi [9]. Anche se ben formalizzata e documentata, la teoria di Lerdahl e Jackendoff presenta indubbiamente alcune lacune e i problemi implementativi da risolvere non sono pochi, poiché molti aspetti non individuabili attraverso la semplice lettura del testo, sono lasciati piuttosto sul vago. Uno di questi riguarda per esempio alcuni dei criteri per i processi di riduzione gerarchica, a livello armonico, di un brano musicale. Vengono introdotti infatti concetti come quelli di vicinanza armonica senza specificarne peraltro una definizione precisa o comunque dare indicazioni per l'effettivo utilizzo ai fini della scelta degli elementi armonici da ridurre.

Uno degli aspetti caratterizzanti della nostra ricerca è quello di aver realizzato un modello che prevede l'interazione tra le procedure di riduzione gerarchica e l'individuazione degli spazi tonali del brano mediante l'esame di valori di stabilità, ottenuti seguendo e modificando un recente studio dello stesso Lerdahl [10]. Egli propone un modello di "spazio tonale" le cui particolarità, come dice lo stesso autore, stanno nel fatto che tratta altezze, accordi e regioni tonali nell'ambito di una medesima struttura. E lo scopo del lavoro è giusto quello di descrivere le relazioni armoniche tra queste tre categorie di elementi in termini delle cosiddette condizioni di prossimità. Sia il termine "prossimità" che "stabilità" denotano a grandi linee lo stesso concetto: esprimono infatti in maniera numerica il concetto di "vicinanza" (e quindi di distanza) di una nota, di un accordo o di una regione rispetto ai rispettivi fondamentali, quali la tonica, l'accordo di primo grado o la tonalità di impianto del brano. Con l'attribuzione dei valori di stabilità è possibile costruire una gerarchia tra eventi musicali distinti, che si differenziano l'un l'altro per vicinanza o lontananza armonica.

Un metodo per il riconoscimento delle modulazioni può essere progettato basandosi su considerazioni che coinvolgono i valori di stabilità attribuiti agli accordi di un brano: il modello di spazio

tonale proposto da Lerdahl può a sua volta essere esteso e modificato ottenendo, dopo una prima fase di modellizzazione della stabilità, una sequenza completa dei valori attribuiti agli accordi. L'esame della sequenza e del suo andamento può dare informazioni riguardo la presenza di una eventuale modulazione.

L'approccio computazionale risulta indispensabile in quanto sarebbe praticamente impensabile costruire manualmente e in poco tempo decine di spazi tonali, confrontare note di accordi, calcolare valori di stabilità, loro medie e decidere infine sulla presenza o meno di una modulazione.

Sia la GTTM che gli ultimi ampliamenti appena descritti costituiscono la base teorica di un sistema di analisi armonica basato su tecniche ibride procedurali e di rappresentazione della conoscenza.

Uno dei più significativi aspetti della GTTM, dal punto di vista armonico, è la riduzione dell'intervallo temporale (*time span reduction*). L'analisi del brano consiste in una progressiva semplificazione dove ad ogni passo eventi meno importanti sono cancellati a favore di altri strutturalmente più importanti (vertici o teste della riduzione) fino a far emergere una sorta di ossatura del pezzo.

Nella riduzione dell'intervallo temporale i domini delle elaborazioni armoniche e melodiche all'interno dei quali scegliere gli

eventi sono definiti dal contesto ritmico delle strutture metriche e di raggruppamento. Ogni linea successiva dell'analisi è il risultato di una cancellazione di eventi relativamente meno importanti della linea superiore. Il procedimento è ripetuto a partire dal livello delle note di durata minima fino al livello che raggruppa l'intero frammento. Se le riduzioni analitiche sono corrette, ogni livello risulta acusticamente una semplificazione naturale della linea precedente. Le regole che permettono la riduzione dell'intervallo temporale a partire dal livello superficiale possono essere schematicamente divise in due parti. La prima parte si fa carico di derivare i domini entro i quali effettuare le riduzioni partendo dalle unità metriche più piccole e salendo verso i domini più grandi. La seconda parte contiene le regole che individuano all'interno del dominio gli elementi funzionanti da testa o vertice della riduzione (elemento gerarchico più importante).

Vediamo quali sono i fattori che determinano la scelta di un elemento come vertice a partire da domini, gli intervalli temporali, contenenti due o più eventi. Ognuno dei fattori non è di per sé determinante per la scelta ma contribuisce in una certa misura alla decisione complessiva. Inizialmente si tiene conto della posizione metrica: un elemento metricamente più forte (in relazione alla sua posizione interna

alla battuta musicale) risulterà preferibile come testa di riduzione. Altri due importanti fattori hanno invece a che fare con le proprietà "tonali" degli eventi: la consonanza armonica intrinseca e la stabilità rispetto alla tonica. Vengono naturalmente preferiti come vertici eventi relativamente consonanti e con un grado di stabilità maggiore. Anche la condotta melodica delle voci viene presa in considerazione a favore di riduzioni che conducano a profili melodici il più possibile stabili.

Man mano che ci si allontana dal livello superficiale aumentano gradatamente di importanza altri fattori legati soprattutto a considerazioni di tipo macro-strutturale. In particolare assumono rilevanza nella riduzione gli eventi che fanno parte di un inizio o di una "fine strutturale" (per esempio le cadenze); solitamente il movimento di una frase musicale si articola proprio tra questi due punti strutturalmente importanti e di relativa stabilità formale.

Devono anche essere inserite regole per il controllo di certe condizioni di parallelismo che si possono verificare sia in senso melodico che armonico tra eventi o coppie di eventi adiacenti.

Struttura del sistema esperto

Il sistema è articolato in diversi blocchi, rappresentanti le diverse

funzioni logiche del processo di analisi armonica della partitura:

1. CODIFICA E ANALISI LESSICALE DELLA PARTITURA
2. RICONOSCIMENTO DEGLI ACCORDI
3. CALCOLO DEI VALORI DI STABILITÀ
4. RICONOSCIMENTO DELLE CADENZE
5. RICONOSCIMENTO DELLE MODULAZIONI
6. PRIMO LIVELLO DI RIDUZIONE
-
- i. N^{MO} LIVELLO DI RIDUZIONE

Inizialmente il sistema analitico prevede un'indagine armonico-lessicale. Il brano viene codificato secondo una divisione in fette di tempo (*slice*), ciascuna comprendente un accordo, che sono riconosciute ed etichettate secondo le loro proprietà armoniche. Dopo una parte dedicata alla determinazione degli spazi tonali, si procede all'individuazione dei punti cadenzali e delle modulazioni, completando la rappresentazione strutturale delle "frasi armoniche". Dopo questo tipo di analisi superficiale si iniziano a distinguere i vari livelli gerarchici dell'armonia secondo gli schemi di riduzione, fino alla completa evidenziazione della struttura del brano.

Entrando nel dettaglio, lo *slice* è una lista di sei campi contenenti le note dell'accordo rappresentate in notazione anglosassone con il sistema nome-alterazione-ottava, la durata dell'accordo, la

dinamica dell'accordo, eventuali segni dinamici o interpretativi ed infine il numero di battuta.

Questa rappresentazione viene via via arricchita automaticamente: al termine del secondo blocco vengono aggiunte le informazioni derivanti dal riconoscimento del tipo di accordo rappresentato.

Per questo problema abbiamo scelto una soluzione classica, intuitivamente semplice: si basa su un algoritmo che calcola gli intervalli tra tutte le note dell'accordo, ordina le stesse per terze e, in base alla successione delle terze, individua il tipo di accordo e gli attribuisce un nome. Vengono riconosciute triadi maggiori e minori, diminuite ed eccedenti, più le precedenti con la presenza di settime e none. Una volta conosciuto il nome dell'accordo si possono stabilire alcune ulteriori sue proprietà come l'appartenenza o meno alla tonalità di base del brano.

Come detto precedentemente la teoria originaria del *tonal pitch space* può essere modificata ed estesa e soprattutto usata in modi e per scopi abbastanza diversi da quelli canonici: è possibile formulare un algoritmo per la rilevazione delle eventuali modulazioni di tonalità attraverso l'esame dei singoli valori di stabilità e delle loro sequenze.

Il metodo utilizzato per il tentativo di individuazione delle modulazioni fa uso sia delle informazioni sulla stabilità relativa degli accordi che di alcuni dati

globali del brano, tra i quali i punti cadenzali. Dopo la fase di modellizzazione della stabilità si ha a disposizione la sequenza completa dei valori attribuiti agli accordi. Ripetute costruzioni sperimentali degli spazi tonali hanno reso possibile l'accertamento di un primo valore di soglia, per la stabilità di un accordo, al di sopra del quale l'accordo stesso può essere considerato come l'inizio di una modulazione. L'ipotesi sulla presenza della modulazione deve essere corroborata da risultati derivanti dall'analisi delle stabilità successive. Se gli accordi successivi, fino al raggiungimento di un punto cadenzale, continuano infatti questa tendenza al superamento della soglia, si potrà affermare con una buone dose di attendibilità che siamo in presenza di una modulazione. Tale modulazione avrà come nuovo accordo di tonica quello di risoluzione della cadenza indicata come limite del campo di esame. L'elenco dei punti cadenzali fornisce informazioni sulla struttura del brano alla parte di rilevamento delle modulazioni e di esecuzione dei livelli di riduzione gerarchica: abbiamo sviluppato un insieme di regole atte alla loro individuazione. Tali regole effettuano dei test riguardanti tre accordi: i due cadenzanti e quello precedente. Prendono in esame il modo degli accordi, le loro posizioni, eventuali segni tipo corone, il profilo melodico della linea del basso,

oltre ad altri fattori che escludono configurazioni derivate da condizioni simili a quelle cadenzali. Le regole sono organizzate secondo una gerarchia di attivazione che consente l'attribuzione del punto cadenzale con diversi gradi di attendibilità. Effettuare invece il primo livello di riduzione significa esprimere una gerarchia tra particolari eventi musicali interni ad una fissata unità ritmica, il *time span*. Nel livello iniziale i fattori che entrano in gioco per la determinazione della testa sono prevalentemente di natura metrica e/o di relazione armonica. A questo livello vengono ridotte, per esempio, le appoggiature, le note di passaggio, le note di volta e gli altri eventi di questa classe. Per confrontare gerarchicamente la consonanza di due accordi abbiamo introdotto una forma di classificazione deducibile dal tipo dell'accordo. La classificazione usata è una nostra variante di quella introdotta da Maxwell nel suo sistema [4] e prevede la seguente suddivisione degli accordi:

- 1) triadi maggiori e minori;
- 2) triadi diminute, maggiori e minori di settima;
- 3) triadi aumentate;
- 4) altre triadi.

All'accordo viene assegnato un valore corrispondente al numero della sua classe. Esiste anche una categoria intermedia tra la prima e la seconda, dove viene collocato esclusivamente l'accordo di settima di dominante della tona-

lità. Agli accordi che hanno una configurazione di primo o secondo rivolto viene aggiunto mezzo punto per compensare la "piccola perdita" di consonanza. Ad ogni regola che governa i processi di riduzione viene assegnato un fattore di certezza che ne contraddistingue il peso nella determinazione dei vertici. Questi pesi possono essere cambiati dall'analista per controllare l'influenza progressiva dei fattori in gioco o la predominanza di uno di essi su gli altri.

In particolare, per la riduzione si opera a livello dell'unità ritmica più piccola del brano preferendo come vertici accordi che risultano più consonanti, che sono più "vicini" alla tonica (tramite i valori di stabilità) e che cadono inoltre su una posizione metrica forte. Ulteriore consistenza alla riduzione viene data da un controllo sulle altezze più acute della melodia e su quelle più gravi della linea del basso. Ognuno dei fattori presi in esame contribuisce alla determinazione finale dell'accordo da eliminare in misura variabile ed eventualmente modificabile da parte dell'analista. Proprio la possibilità di intervenire su questi fattori, varriandone il peso relativo, può essere veicolo di una interessante sperimentazione circa la diversa importanza degli aspetti sottostanti i processi di riduzione e sulla loro interazione.

Il procedimento di riduzione prosegue per un numero di livelli non determinabile a priori.

La sua lunghezza dipende, oltre che dai valori minimi di durata degli eventi presenti, dalla complessità del brano in esame.

Nei livelli successivi al primo rimangono in azione tutti i fattori elencati precedentemente ma ne iniziano ad entrare in gioco altri che obbligano la lettura contemporanea di un numero di accordi superiore a due, per esempio quaterne di eventi. Ed è proprio la scelta dell'ampiezza del *time-span* uno degli aspetti più difficolosi di questa sezione analitica. Le regole devono tener conto della presenza di anacrusi, di condizioni di fine ed inizio battuta, dei battiti metrici interni alle battute e di molti altri parametri. Da notare che il concetto stesso di battuta perde di significato man mano che ci si allontana dal livello superficiale; in genere i livelli dal terzo in poi non ne fanno praticamente più uso.

La scelta degli elementi da conservare al livello via via in oggetto può essere regolata da considerazioni riguardanti anche certe caratteristiche di continuità. Per esempio se all'interno della quaterna gli accordi 1 e 4 hanno le voci del soprano che distano un tono o semitono, oppure quelle del basso che distano lo stesso intervallo, si preferisce ridurre gli altri e lasciare questi, o quantomeno attribuire loro una probabilità alta di non riduzione, valutabile successivamente assieme ai dati provenienti dai confronti degli altri fattori. Regole

apposite attribuiscono inoltre una buona probabilità di essere vertice di riduzione ad elementi appartenenti agli "inizi o fini strutturali" di frasi musicali. Essendo questi ultimi alcuni degli elementi che determinano l'ossatura portante di una frase o di un brano musicale, è naturale pensare che anche gli eventi che contribuiscono alla loro funzione debbano subire un trattamento particolare, ritardando il più possibile la loro riduzione.

Un altro fattore molto importante da prendere in considerazione è quello basato sulle regole di parallelismo e che può essere efficacemente definito con le parole stesse di Lerdahl e Jackendoff: "se due o più intervalli temporali possono essere costruiti come paralleli da un punto di vista del motivo o del ritmo, allora assegna loro dei vertici di riduzione paralleli" [1]. Naturalmente si tratta di una definizione a carattere generale dato che nella pratica la verifica delle condizioni di parallelismo è assai difficoltosa e dispendiosa a livello computazionale. Possono infatti sorgere numerosi casi diversi quali la presenza in *span* contigui di progressioni di intervalli uguali oppure di rapporti intervallari uguali in relazione al grado degli accordi e altri casi. Questa non ben specificità riguardo al parallelismo è forse uno dei punti deboli della teoria generativa della musica tonale e di fatto costituisce un facile terreno di sperimentazione dell'ana-

lisi musicale, soprattutto dal punto di vista cognitivo.

L'ultimo livello di riduzione, il più basso nella notazione musicale, la radice dell'albero nella notazione linguistica, è rappresentato dall'unico accordo di tonica della tonalità che, banalmente, coincide spesso con l'ultimo accordo del brano.

Implementazione e sperimentazione

Per quanto concerne la realizzazione, il sistema è costituito da una base di conoscenza formata da regole per la rappresentazione dei dati e del modello analitico; esse vengono attivate, tramite un motore inferenziale che usa tecniche di *forward* e *backward chaining* e un insieme di strutture di controllo che rappresentano la strategia analitica (Focus Control Blocks). Al sistema è associata una parte procedurale, scritta in *Pascal*, che si fa carico di risolvere i problemi di carattere microstrutturale quali lo svolgimento di funzioni prettamente algoritmiche e lo scambio dei dati con l'esterno.

Tornando alla struttura del sistema logico, la divisione gerarchica dei compiti di analisi può essere evidenziata dall'albero degli FCB (Fig. 1). Ognuno di essi rappresenta un insieme logico di operazioni e contribuisce a creare una struttura ad albero i cui nodi vengono percorsi a cicli per seguire il ragionamento analitico e portare in evidenza i risultati.

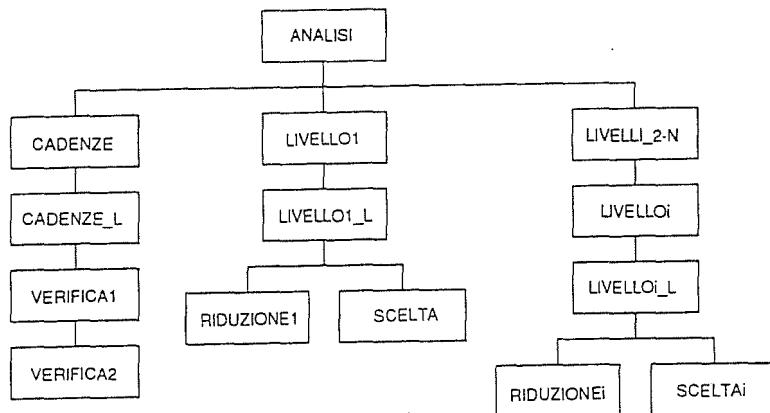


Fig.1. Albero di attivazione della base di conoscenza.

Il FCB ANALISI controlla lo svolgimento complessivo delle operazioni e, dopo l'acquisizione di dati generali del brano, stabilisce la lista dei movimenti forti del brano per mezzo di regole specifiche. Attiva poi sequenzialmente i suoi nodi figli, corrispondenti alle operazioni analitiche seguenti. Ad ogni livello dell'albero sono attivate procedure esterne per lo scambio dei dati intermedi e per il passaggio dei risultati ottenuti.

Il FCB CADENZE attiva la parte per l'individuazione delle cadenze, il FCB LIVELLO₁ provvede al primo livello di riduzione mentre il FCB LIVELLO_N ripete il procedimento di riduzione dal secondo livello all'ultimo.

Ogni gruppo di FCB necessita di informazioni particolari sul brano analizzato. Abbiamo scelto quindi di non rappresentare gli *slice* integralmente e con modalità identiche nel corso della con-

sultazione, ma di predisporre invece un sistema capace di cambiare l'insieme dei dati di volta in volta disponibili per le scelte guidate dalle regole. Si tratta di una forma di rappresentazione "variabile", nel senso che per ogni sottoproblema si fa riferimento ad uno specifico sottosistema delle informazioni disponibili, organizzate in strutture a lista.

Il sistema di analisi armonica accetta in ingresso la codifica alfano-merica del brano musicale da analizzare e fornisce, al termine delle procedure analitiche, un'analisi armonico-strutturale del brano stesso espressa in termini dei livelli di riduzione. In mezzo a questi due estremi sono disponibili comunque tutta una serie di risultati intermedi, altrettanto degni di interesse da un punto di vista musicologico e importanti anche ai fini di un costante controllo sull'esecuzione del processo analitico. Questi

risultati, che potremmo definire parziali, sono rappresentati da:

- 1) riconoscimento degli accordi, espresso come etichettatura degli stessi e verificabile all'interno di un file di controllo apposito;
- 2) descrizione dei punti cadenzali, riscontrabile all'interno della base di conoscenza mediante stampa della lista specifica;
- 3) descrizione delle sezioni di modulazione armonica, riscontrabile mediante esame del file contenente le informazioni sugli *slice*.

Un risultato indiretto utile ai fini del processo di integrazione, verifica e ampliamento della teoria è costituito dalla traccia del ragionamento (*trace file*) disponibile in diversi livelli di dettaglio come risultato di un processo di consultazione della base di conoscenza. Possono essere evidenziati il passaggio di controllo tra i blocchi, la ricerca di valori e il loro assegnamento ai parametri, la risoluzione dei vincoli e soprattutto la prova e l'attivazione delle regole, il loro stato ad ogni istante e altre informazioni secondarie.

Per quanto riguarda la consultazione, ogni risultato intermedio che compare sullo schermo può essere modificato dall'analista musicale semplicemente sovrascrivendo valori diversi da quelli presentati. Egli può inoltre, come detto, intervenire sui pesi di certe regole di riduzione modificandone i valori da tastiera.

Inizialmente i test, sia delle fasi di pre-elaborazione che poi delle fasi inferenziali, sono stati condotti su brani relativamente semplici, tipicamente appartenenti al repertorio barocco, proseguendo poi con un progressivo aumento di complessità dei pezzi in analisi. Man mano che il grado di complessità musicale sale sono state tenute in considerazione, con l'aggiunta di regole e con modifiche alle parti procedurali, sempre nuove variabili e nuovi punti di vista per la risoluzione dei problemi.

Il processo di retroazione descritto ha messo in luce, già in alcuni dei primi esempi di analisi, alcune discrepanze con i risultati manuali ottenuti dagli stessi Lerdahl e Jackendoff: si tratta soprattutto di differenze nei risultati delle riduzioni, differenze che non devono essere interpretate come errori nella formalizzazione della teoria generativa ma anzi come espressione di una diversa interpretazione di alcune casistiche musicali. Sarà interessante verificare, sia mediante una discussione con gli autori della GTTM che con il confronto con altri impianti teorici, le motivazioni e le conclusioni da trarre nei confronti di questo genere di risultati.

Conclusioni

L'alto grado di complessità dei fenomeni musicali impone delle forme di realizzazione che siano adeguate e che, a fini di correttezza e completezza, affrontino i

problemi sotto un numero di analizzazioni sufficientemente ampio [6]. Per questi motivi appare sempre più interessante la possibilità di integrare in maniera coerente, formalismi di rappresentazione della conoscenza musicale di differente natura, all'interno dei cosiddetti sistemi ibridi [11].

L'esempio di sistema esperto da noi proposto si colloca in una tale dimensione. Con questo lavoro abbiamo cercato di ampliare, integrare e verificare gli assunti della teoria generativa della musica tonale proposta dagli studiosi americani Lerdahl e Jackendoff, e della teoria sulle regioni tonali dello stesso Lerdahl. Laddove la trattazione di certi aspetti analitico-musicali non fosse stata precedentemente contemplata, sono state cercate nuove specifiche soluzioni teoriche. Pertanto anche da un punto di vista teorico-musicale la possibilità di integrare differenti approcci appare come una via foriera di interessanti sviluppi. Ed è proprio grazie alle nuove tecniche di intelligenza artificiale che diventano fattibili tali forme di integrazione e di verifica dei risultati.

Bibliografia

- [1] F. Lerdahl, R. Jackendoff: *A Generative Theory of Tonal Music*, Cambridge Mass., The MIT Press, 1983.
- [2] F. Lerdahl, R. Jackendoff: "An Overview of Hierarchical Structure in Music", *Music Perception*, Vol.1, N.2, 1984.
- [3] L. Camilleri, F. Carreras, C. Duranti: "An Expert System Prototype for the Study of Musical Segmentation", *Interface*, Vol.19, N.2-3, 1990.
- [4] J.H. Maxwell: "An Expert System for Harmonic Analysis of Tonal Music", Proc. of the First International Workshop on Artificial Intelligence and Music, AAAI Press, 1988.
- [5] K. Ebcioğlu: "An Expert System for Harmonizing Four-part Chorales", *Computer Music Journal*, Vol.12, N.3, 1988.
- [6] F. Giomi: "Analisi musicale e intelligenza artificiale. Problemi, metodi ed esempi", Eunomio, N.20, 1993.
- [7] M.J. Baker: "An Artificial Intelligence Approach to Musical Grouping Analysis", *Contemporary Music Review*, Vol.3, N.1, 1989.
- [8] D. Scarborough et al.: "An Expert System for Music Perception", Proc. of the First International Workshop on Artificial Intelligence and Music, AAAI Press, 1988.
- [9] F. Holm: "Machine Tongues XIV: CSP-Communicating Sequential Processes", *Computer Music Journal*, Vol.16, N.1, 1992.
- [10] F. Lerdahl: "Tonal Pitch Space", *Music Perception*, Vol.5, N.3, 1988.
- [11] A. Camurri: "On the Role of Artificial Intelligence in Music Research", *Interface* Vol.19, N.2-3, 1990.

FORMALIZZAZIONE DI STRUTTURE GENERATIVE ALL'INTERNO DE "LA SAGRA DELLA PRIMAVERA"

Adriano De Matteis, Goffredo Haus

Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39
I-20135 Milano (Italia)
fax +39 2 55006373
e-mail: music@imiucca.csi.unimi.it

Abstract

In this paper we show how we can formally describe Generative Structures (GS) of musical pieces. We consider as a particular case study "The Rite of Spring" by I. Stravinsky.

The first goal of our research has been the identification of relationships among Music Objects (MOs) belonging to the piano duet or the orchestral score or both. We have defined two kind of GSs: Essential Generative Structures (EGSs) and Additional Generative Structures (AGSs). We call EGS a structure that determines the development of music material only in a temporal sense (Horizontal development), while the concept of AGS concerns with the orchestration of EGSs. We think that orchestration must be thought not only at a timbric but also harmonic level by superimposition of new generated material (Vertical development). We can generally say that the same EGS occurs both in the piano reduction and in the orchestral score, while AGS only in the latter.

Our formalization has been made by means of an "ad hoc" arrangement of hierarchical Petri Nets (PNs) and a music algebra; music objects are associated to places; music transformations are described by algorithms associated to transitions and are coded by expressions which are based on the set of operators and the syntactic rules we have defined.

The models we have produced have been implemented by the ScoreSynth software module of the Intelligent Music Workstation.

This research has been partially supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

Introduzione.

Nel presente articolo mostriamo come sia possibile descrivere formalmente Strutture Generative (GS) di brani musicali. Consideriamo come caso particolare di studio "La Sagra della Primavera" di I. Stravinsky.

Il concetto di Oggetto Musicale (MO) ha un ruolo fondamentale nella nostra ricerca; un MO può essere qualunque cosa abbia un significato musicale e che possiamo pensare come entità, sia semplice che complessa, sia astratta che dettagliata, un'entità con un nome e che abbia qualche relazione con altri oggetti musicali.

Il primo scopo della nostra ricerca è stato l'identificazione delle relazioni tra MO appartenenti alla riduzione per pianoforte a quattro mani [2] o alla partitura orchestrale [3] o ad entrambe.

Abbiamo definito due tipi di GS:

- 1) Essenziali (EGS);
- 2) Aggiuntive (AGS).

Definiamo EGS una struttura che determina lo sviluppo del materiale musicale esclusivamente da un punto di vista temporale (Sviluppo orizzontale). Nella musica di Stravinsky questo può significare l'opposizione e/o la giustapposizione di due o più pattern melodico-ritmici periodici e variabili, la sincronizzazione e l'incastro di periodi per ottenere effetti di asimmetria, e così via.

Il concetto di AGS riguarda invece l'orchestrazione di EGS. Orchestrione che deve essere pensata in senso generale come mezzo per generare materiale nuovo, non solo a livello timbrico, ma anche armonico, mediante sovrapposizione di nuove generazioni (Sviluppo Verticale).

Come idea generale possiamo dire che una stessa EGS si trova sia nella riduzione per piano che nella versione orchestrale corrispondente, mentre una AGS solo in quest'ultima.

La nostra formalizzazione è stata fatta mediante un opportuno adattamento delle Reti di Petri (PN), e un'algebra musicale [1]; gli MO sono associati ai posti della rete; le trasformazioni musicali vengono descritte mediante algoritmi associati alle transizioni e codificati da espressioni che sono basate sullo insieme di operatori e sulle regole sintattiche che abbiamo opportunamente definito. Questo approccio ci permette di descrivere gli oggetti musicali e le loro trasformazioni a vari livelli di rappresentazione.

Una EGS è formalizzata da una PN in cui i MO d'ingresso rappresentano idee melodico-ritmiche, e la PN determina lo sviluppo del materiale generato solo in senso orizzontale. Una AGS è una rete che permette il raffinamento di una EGS corrispondente. Quindi una AGS realizza un morfismo tra PN che, seguendo (pensata come)

una procedura top-down, conserva le strutture temporali stabilite dalla rete EGS corrispondente.

Un caso esemplare.

Mostriamo ora attraverso un esempio caratteristico in che modo si sviluppa la nostra ricerca. L'esempio ci permetterà di mettere in evidenza i seguenti punti, che sono anche gli scopi generali della nostra ricerca:

- 1) è possibile trovare un insieme di PN che funziona su un buon numero di brani;
- 2) un insieme di MO e un numero ridotto di queste PN può descrivere parzialmente o totalmente, lo sviluppo del materiale musicale di una composizione;
- 3) si può passare attraverso un processo di raffinamento, caratteristico delle PN, dalla partitura per pianoforte a quella corrispondente per orchestra;
- 4) questa formalizzazione è intimamente connessa al pensiero e alla pratica musicale, in quanto conserva e mostra le strutture nascoste delle partiture musicali; in più risulta sufficientemente maneggevole.

La partitura generata nello esempio riportato è il modello della partitura dal numero 181 al numero 185 ed è il risultato di una serie di trasformazioni di un unico MO che chiameremo SOGGETTO.

Da esso deriva un gruppo soggetto-risposta S-R, a intervallo di quarta superiore,

parallelamente al quale si sviluppa un'altro gruppo soggetto-risposta, che comporta lievi differenze cromatiche, S1-R1, anch'esso derivato da SOGGETTO, presente nella partitura orchestrale, ma non in quella pianistica.

Figura 1.

Inizialmente viene esposta una serie di variazioni del soggetto, variazioni che riguardano la prima nota (testa del soggetto): queste variazioni ne alterano sia le ripetizioni sia la durata. Questa serie di variazioni è stata chiamata S. Ad esempio, nelle prime sei battute la testa del soggetto viene ripetuta cinque volte e, tranne la prima, le altre ripetizioni durano il doppio. Quindi il soggetto viene riproposto una quarta sopra e subisce ancora delle modificazioni della testa come visto precedentemente. Ad esempio nella battuta 15, la testa viene ripetuta tre volte, senza modificazioni di durata. Questa serie di variazioni è stata chiamata R. Un analogo trattamento è riservato al gruppo S1-R1, che procede parallelamente al gruppo S-R.



Figura 2: partitura completa del modello.

In più il soggetto subisce delle alterazioni cromatiche. Nella Figura 2 abbiamo indicato, per semplicità, con S, R, S1, R1, soltanto la prima delle ripetizioni di ciascun gruppo di variazioni; inoltre il gruppo S1-R1 è rappresentato trasposto di un'ottava inferiore.

Vediamo come procede il modello mediante PN. In Figura 3 abbiamo la rete al livello più alto.
Il soggetto è associato al posto SOGGETTO. Inizialmente il posto INIZIO ha una marca mentre gli altri non ne hanno. Dunque la transizione A può scattare.

Alla transizione A è associato l'algoritmo:

P : 1, \$, [SOGGETTO, 1], ?

che ha come effetto quello di caricare il soggetto.

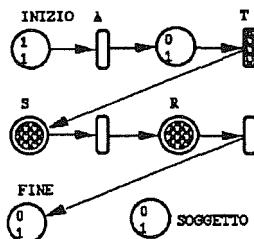


Figura 3: rete S-R.

Se alla transizione T non è associata una sottorete, T scatta e si passa all'esecuzione del gruppo sequenziale S-R. La transizione T realizza, quando è invece associata ad una opportuna sottorete, il raffinamento di cui si è parlato e che ci riserviamo di esaminare in seguito. Ai posti S e R è associata una stessa sottorete chiamata MACRO che realizza l'esposizione del soggetto, quando è chiamata da S, e della risposta, quando è chiamata da R con le rispettive variazioni che abbiamo già evidenziato. La macro di cui si parla è rappresentata dalla Figura 4.

Questa rete riceve in ingresso da S o da R il soggetto non ancora modificato. Lo scatto della transizione DISTRIBUZIONE determina la creazione di tre

copie del soggetto che vengono associate ai posti COPIA1, COPIA2, COPIA3. Quindi nelle transizioni indicate con TRASF vengono realizzate le modifiche necessarie.

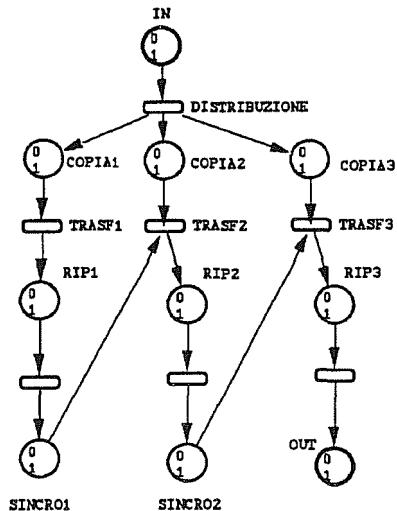


Figura 4: rete MACRO.

Ad esempio nella lista parametri di chiamata del posto S troviamo:

TRASF1 :

{ M : 1, 1, 5
ripeti la prima nota cinque volte
D : 2, 4, ?*2
D : \$, \$, ?*2 }
raddoppia le durate dalla seconda
alla quarta e poi dell'ultima nota

In questo modo la serie di ripetizioni del soggetto conterrà le variazioni indicate. Osserviamo che la transizione TRASF1 è l'unica a poter scattare in quanto TRASF2 e TRASF3 hanno in

ingresso i posti SINCRO1 e SINCRO2 che non hanno marche. Al posto RIP1 è associata l'esecuzione effettiva del risultato delle trasformazioni avvenute in TRASF1.

Eseguita questa prima ripetizione il posto SINCRO1 avrà una marca. Dunque la transizione TRASF2 può scattare e lo algoritmo ad essa associato generare una nuova ripetizione del soggetto con le variazioni necessarie e così via. Osserviamo che il posto SINCRO1 ha una funzione logica: indica cioè la fine della prima esposizione e quindi che è possibile iniziare un'altra. Analogamente si sviluppano le altre ripetizioni. Infine il posto OUT ci riporta nella rete principale S-R. Il medesimo procedimento è associato, con le opportune modifiche, al posto R. Ad esempio, nei parametri di chiamata della sottorete MACRO nel posto R:

DISTRIBUZIONE:
{P:1, \$, [SOGGETTO, 1], ?+5}
trasponi il soggetto una quarta sopra

TRASF1:
{M:1, 1, 3}
ripeti la prima nota tre volte

Fin qui il modello rappresenta lo sviluppo della partitura per pianoforte. Vediamo come si arriva alla versione orchestrale con una AGS. Questa AGS è una sottorete associata alla transizione T ed è rappresentata in Figura 5. La transizione IN determina un parallelismo perché l'effetto di un

suo scatto è di porre una marca nel posto P e una nel posto S1. Quindi può iniziare l'esecuzione del gruppo S1-R1 e parallelamente quella del gruppo S-R mediante lo scatto della transizione OUT che ci riporta nella rete principale S-R. Per i posti S1 e R1 valgono discorsi analoghi a quelli fatti in precedenza per S e R.

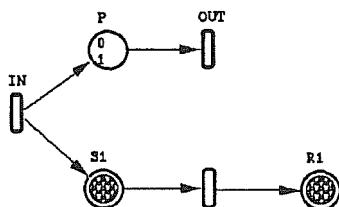


Figura 5: RETE.

Questo raffinamento ci porta alla versione orchestrale.

Sottolineamo il fatto che ad S, S1, R, R1 è associata la stessa sottorete MACRO.

Risulta quindi che da un solo MO (SOGGETTO) e con l'uso di poche reti, tra cui una in particolare (MACRO) chiamata più volte è possibile ricavare ampie parti di una partitura. In più mediante l'uso di una sola rete ausiliaria è possibile orchestrare la precedente partitura.

Conclusioni.

Naturalmente, sono possibili i più svariati approcci descrittivi. Abbiamo seguito criteri di analisi che sembravano particolarmente adatti a descrivere le strutture

generative all'interno de "La Sagra della Primavera" e ne abbiamo riportato, per brevità, un solo campione esemplare caratterizzante il metodo.

La ricerca può essere utile non solo con finalità di analisi musicologica, ma anche come strumento per illustrare le possibilità di una simile formalizzazione nella composizione di nuovi brani, ad esempio utilizzando, tutte o in parte, le strutture generative individuate e descritte nel corso della ricerca.

I modelli che abbiamo prodotto sono stati implementati con il modulo software ScoreSynth della Stazione di Lavoro Musicale Intelligente e così possiamo avere tanto partiture quanto esecuzioni musicali dirette come riscontro dei nostri modelli nelle diverse varianti formulate.

Riferimenti bibliografici.

[1] G. Haus, A. Sametti: "SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra", in "Readings in Computer Generated Music", D. Baggi Ed., pp.53-78, IEEE Computer Society Press, 1992.

[2] Stravinsky, I., "The Rite of Spring - Reduction for Piano Duet", Boosey & Hawkes, London, 1947.

[3] Stravinsky, I., "The Rite of Spring", Boosey & Hawkes, London, 1967.

USING SELF-AFFINE FRACTAL CODING TO MODEL MUSICAL SEQUENCES

Bruno Fagarezzi^{**} & Massimo Sebastiani*

* D.E.I. Dipartimento di Elettronica e Informatica
University of Padua, via Gradenigo 6/a
35131 Padova (Italy)
tel +39(0)498287500 - fax +39(0)498287699
E-mail adtpoli@ipduniv.bitnet

° C.S.C. Centro di Sonologia Computazionale
University of Padua, via S.Francesco 11
35121 Padova (Italy)
tel +39(0)498283757 - fax +39(0)498283733
E-mail music01@ipdunivx.it

Abstract

Starting from the interpolation algorithm already described in the paper presented at IX CIM [1], we have developed a new method to compress sound events. In this paper such method is presented and different implementation of it are suggested in order to allow musicians and researchers to apply the method in the best way for fitting their own exigencies.

Interpolazione

Vale la pena richiamare brevemente l'interpolazione tramite IFS che si presta alla generazione di segnali e funzioni per inviluppi e melodie monofoniche mentre risulta poco adatta alla

polifonia ed alla rappresentazione del tempo e del ritmo.

Le funzioni vengono generate fissando: i punti di partenza tra i quali si intende interpolare, la dimensione frattale per ogni intervallo e, nel caso multidimensionale, i parametri di correlazione tra una funzione e l'altra. Ogni intervallo è sottoposto ad una trasformazione affine che mappa tutto il dominio nello stesso intervallo. Essa è del tipo:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

E' importante sottolineare che per ogni intervallo esiste una sola funzione che lo ammette come codominio: in sostanza i codomini delle funzioni, che pure hanno lo stesso dominio, non si sovrappongono.

Compressione

La compressione tramite IFS si presta molto bene all'approssimazione di parametri fisici così come alla rappresentazione delle microvariazioni timbriche ma non tanto alla compressione di melodie. Essa si ottiene direttamente dall'algoritmo relativo all'interpolazione basandosi sul fatto che la funzione da approssimare è un punto fisso (cioè un attrattore) della trasformazione:

$$T = \bigcup_{i=1}^N w_i$$

dove w_i sono singole trasformazioni affini. Per ottenere la compressione è sufficiente trovare le w_i che mappano parte della funzione in altre sue parti con eventuali sovrapposizioni. Inoltre, una volta fissati dominio e codominio, per trovare la w_i più appropriata si possono stabilire i gradi di libertà rimanenti minimizzando l'errore di approssimazione col metodo dei minimi quadrati.

Il problema si riduce quindi a trovare, per ogni intervallo (codominio), il dominio che minimizza l'errore ed a ridurre l'intervallo considerato nel caso in cui l'errore ecceda il massimo consentito. Il collage theorem [2] garantisce che l'errore finale E commesso nell'approssimazione è:

$$E < \frac{e}{1-s}$$

dove e è l'errore calcolato ai minimi quadrati ed s è il fattore di contrazione della mappa che deve essere sempre minore o uguale ad 1 (mappe contrattive in cui i domini sono più ampi dei codomini). Si può vedere pertanto che, a differenza di quanto succede nel caso dell'interpolazione, nella compressione siamo liberi di scegliere i domini come vogliamo ed il tipo di funzione può anche non essere rigorosamente autosimile.

Modello

Questo nuovo modello di IFS permette la rappresentazione di eventi musicali di qualsivoglia complessità. Si utilizza una rappresentazione di eventi costituita da n-uple di valori che possono contenere informazioni riguardanti durata, altezza, timbro oltre alla componente temporale per altro indispensabile.

La grossa differenza rispetto al modello relativo all'interpolazione consiste nella diminuzione della dimensione frattale dell'attrattore che diventa minore di 1 essendo l'insieme puntiforme.

Per il caso bidimensionale le trasformazioni affini sono del tipo:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} a & 0 \\ c & d \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

dove, per quanto riguarda le du-

rate, si può anche usare un artificio:

$$durata_{n+1} = a \cdot durata_n + e$$

in quanto è verosimile che le trasformazioni che agiscono sulla componenete temporale (x) possano interessare anche le durate. La trasformazione affine può essere facilmente estesa al caso n-dimensionale nel modo seguente:

$$\begin{bmatrix} x_1 \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{1n} \\ a_{n1} & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ y_n \end{bmatrix} + \begin{bmatrix} e_1 \\ e_n \end{bmatrix}$$

mentre molto più semplicemente, nel caso in cui non si voglia tener conto delle interazioni tra i vari parametri descrittivi, l'estensione a più dimensioni si può ottenere ripetendo il procedimento relativo al caso bidimensionale per ogni dimensione separatamente.

Risulta evidente che in questo modello l'errore di approssimazione gioca un ruolo determinante.

Approssimazione

Supponiamo di voler approssimare un intervallo in uno spazio n-dimensionale (N-cubo) con un dominio prefissato. Per trovare la w_i che minimizza l'errore di approssimazione si può procedere in più di un modo.

Nel caso bidimensionale si può risolvere il problema col metodo dei minimi quadrati per il quale

il sistema è di facile soluzione in quanto, sotto opportune condizioni, esso si abbassa di grado. Nel caso di spazi di dimensione superiore a due, oltre al metodo dei momenti, la tecnica che sembra fornire i risultati più interessanti è quella relativa agli algoritmi genetici.

Analogamente al caso ad una sola dimensione, l'algoritmo di approssimazione deve provvedere a scegliere i domini ed i codomini adatti (n-cubi) che minimizzano l'errore di trasformazione. Se il numero di punti da approssimare non è elevato, un algoritmo esaustivo di ricerca (con un minimo di 'branch and bound') può essere efficace. La ricerca termina quando si è trovata, per ogni intervallo, una funzione che lo ammette come codominio.

Ci sembra opportuno sottolineare il fatto che non si tratta di una semplice scomposizione in operatori musicali (trasposizioni, inversioni, ecc) in quanto gli eventi vengono sintetizzati con lIFS come limite di una successione di applicazioni di funzioni e per questo motivo essi sono intrinsecamente strutturati sia localmente che globalmente.

Modello con Condensazione

Quando l'approssimazione di un intervallo risulta troppo complessa e di conseguenza il numero di trasformazioni da aggiungere diventa troppo elevato, si può semplificare l'analisi la-

sciando l'intervallo non compresso, cioè privo di funzioni che lo ammettono come codominio, aggiungendo al loro posto una funzione tabellata di identità su tale intervallo ($w(A)=A$).

In questo caso A viene chiamato 'set di condensazione' e può essere scelto in base a criteri strutturali riguardanti la composizione, in modo da rappresentare un tema od una frase da cui derivare il resto mediante trasformazioni.

Risulta peraltro evidente come modificando anche lievemente il set di partenza A si possano ottenere infiniti attrattori che mantengono la stessa struttura.

Il set di condensazione può essere usato sia in una che in n dimensioni.

Modello 'Evolutivo'

Se non si inserisce la coordinata temporale tra quelle da analizzare, si ottiene un insieme di trasformazioni, ciascuna delle quali rappresenta un 'fotogramma' dello stato della composizione in un certo istante (t_1).

Una caratteristica fondamentale dell'IFS è quella di avere un comportamento stabile rispetto alle perturbazioni dei coefficienti delle trasformazioni. Perciò, se si calcola un altro 'fotogramma' in un istante successivo (t_2) utilizzando lo stesso numero di funzioni e gli stessi intervalli come suddivisione in domini e codomini, si può calcolare il

'fotogramma' in un istante intermedio interpolando linearmente i coefficienti ed ottenendo un attrattore che esibisce una morfologia di transizione tra quelli relativi ai due istanti t_1 e t_2 .

Si possono quindi ottenere un numero qualsiasi di 'istanti di metamorfosi' tra una struttura ed un'altra fino a raggiungere un'impressione di continuità compatibilmente con la risoluzione minima.

Infine si può procedere operando direttamente sulle funzioni o sulla struttura nel corso dell'evoluzione temporale: situazione ottimale per il controllo di fasce timbriche o per la sintesi granulare.

Bibliografia

[1] B.Fagarazzi, M.Sebastiani: *"Codifica di eventi sonori e dei relativi parametri formali con il metodo dell'I.F.S."* atti IX C.I.M., pp.314322, AIMI-DIST Publ. 1991.

[2] M.F.Barnsley *"Fractals Everywhere"*, Springer Verlag N.Y. 1989.

ANALISI STATISTICHE NEL RICONOSCIMENTO DEGLI INTERVALLI.

Ugo Merlone

(A.M.E.T.-Conservatorio G.Verdi Torino)
P. zza Galimberti 1 10134 Torino
Tel. 011-3177119

Abstract

With this paper I mean to support a statistical and Quality-Control like approach to the harmonic and melodic interval recognition. The importance of the ear-training for the musician is well known, nevertheless not everybody has these capacities developed, and so a long training is often needed to reach the "pass mark".

Programs dealing with these problems already exist and are available at some Softwarehouse, but with this paper I want to suggest a new approach. As a matter of fact each user successes and failures are recorded, and on these data statistical analyses are built (type I and type II errors, phenomena Pareto's charts). These analyses can be performed on each kind of proposed intervals and are oriented to

achieve causes of failure in the proper recognition.

Moreover it is possible to tailor aimed exercitation on the basis of the analysis results. This kind of methodology is implemented under Ms-Dos on IBM-compatible computer with VGA, Roland MPU401 Midi interface; the source code is written in C++ and versions applied on melodic intervals in the octave and over, harmonic intervals and note recognition with the movable Do method are available.

1 Analisi statistica degli errori.

Con questo lavoro si è pensato di dare una valenza risolutiva alle analisi statistiche normalmente presenti nei numerosi software di ear-training. Si è infatti proceduto a considerare gli errori nel riconoscimento di un determinato

intervallo come dati grezzi per eseguire varie analisi statistiche:

- Errori di I e II specie
- Diagrammi di Pareto

1.1 Errori di I e II specie.

Dalla teoria statistica dei test d'ipotesi si sono presi a prestito due locuzioni molto utili nell'interpretare gli errori che si compiono nel riconoscimento degli intervalli.

Quando si prende in esame un test statistico di un'ipotesi si possono commettere due tipi di errore:

- Errore di I specie: quando si rifiuta un'ipotesi vera
- Errore di II specie: quando si accetta un'ipotesi falsa

Considerando un determinato intervallo i , quando viene proposto un intervallo $x = i$ chiamerò errori di prima specie gli intervalli di risposta $g \neq i$ che vengono detti erroneamente essere i ; e quando viene proposto un intervallo $x' \neq i$, chiamerò errori di seconda specie gli intervalli $g = i$ che sono ovviamente errati.

Formalizzando, siano:

- I : l'insieme degli intervalli considerati

- $x \in I$: l'intervallo da riconoscere
- $g \in I$: l'intervallo risposta
- E^1_i : l'insieme degli errori di prima specie rispetto all'intervallo $i \in I$
- E^2_i : l'insieme degli errori di seconda specie rispetto all'intervallo $i \in I$

Si avrà, considerato un $i \in I$:

$$E^1_i = \{g \in I / g \neq i \wedge x = i\}$$

e

$$E^2_i = \{g \in I / g = i \wedge x \neq i\}$$

1.2 I diagrammi di Pareto.

Nel 1897 l'economista italiano Vilfredo Pareto, prendendo in esame alcuni dati statistici di vari paesi, giunse a concludere che la distribuzione del reddito non è uniforme; una teoria simile è stata esposta, sotto forma di diagramma, dall'economista americano M. C. Lorenz nel 1907. Entrambi gli studiosi hanno evidenziato che la maggior parte della ricchezza è posseduta da un numero molto ristretto di persone. Nel campo del controllo della qualità, J.M. Juran è riuscito ad applicare il metodo del diagramma di Lorenz come strumento per classificare i

problemi primari, pochi ma di notevoli effetti, e problemi secondari, numerosi ma dagli effetti limitati. Mediante questo metodo che ha chiamato analisi di Pareto, ha infatti posto in evidenza che in molti casi la maggior parte dei difetti e dei loro costi è causata da un numero relativamente piccolo di cause.

Vi sono vari tipi di diagrammi di Pareto, quello qui considerato consiste semplicemente in una distribuzione di frequenza (o istogramma) di dati in cui si ha sull'asse delle ascisse il tipo di difetto ordinato per frequenze decrescenti e sulle ordinate la percentuale del difetto.

2 Struttura del software

La struttura del software è molto semplice, una volta caricati i dati sugli errori precedentemente commessi, è possibile procedere a riconoscere:

- una serie di intervalli determinata in modo indipendente dai precedenti errori
- una serie di intervalli determinati con una distribuzione di densità le cui probabilità sono direttamente proporzionali agli errori compiuti

3 Possibili Sviluppi

Pur essendo la versione originale dedicata al riconoscimento degli intervalli melodici ascendenti e discendenti nell'ambito dell'ottava giusta, sono state sviluppate alcune versioni dedicate al riconoscimento degli intervalli armonici nell'ambito di due ottave, e degli intervalli melodici ascendenti sempre nell'ambito di due ottave.

Sono inoltre in corso di sviluppo una versione dedicata al riconoscimento assoluto degli intervalli, una al riconoscimento di figurazioni ritmiche elementari ed un software che permetta di utilizzare i dati sugli errori precedenti per determinare dei semplici dettati melodici.

Bibliografia

- [1] A. Agamenone, *"Teoria Fondamentale della Musica"* Carish, Milano.
- [2] J.K. Galbraith, *"Storia della Economia"* Rizzoli, Milano, 1988.
- [3] H. Kume, *"Metodi statistici per il miglioramento della qualità"* Isedi Petrini, Torino, 1988.
- [4] D. C. Montgomery, *"Statistical Quality Control"*, John Wiley & Sons, New York, 1985.

[5] J. A. Paulos, "*Innumeracy*",
Penguin Books, London, 1990.

[6] S. M. Ross, "*Introduction to probability and statistic for engineers and scientists*" , John Wiley & Sons, New York, 1987.

DEFINIZIONE DI RETI DI PETRI PER L'ANALISI DELLA MUSICA ELETTOACUSTICA

Simonetta Sargentì

Civica Scuola di Musica di Milano
Corso di Porta Vigentina 15, I-20122 Milano tel.58314433

Abstract

This paper discusses how to analyse and represent a piece of electroacoustic music by Petri Nets. The piece is *Resonant Wholes* by Claudio Gabbiani and Simonetta Sargentì, entirely realised with the M.A.R.S. (Musical Audio Research Station -Iris,Bontempi-Farfisa). Generally, electroacoustic music cannot be represented by a traditional score; Petri Nets is a modelisation that make possible to describe electroacoustic music events and their relations. One can create many kinds of nets at differents levels: 1) for representing the thematic structure of the piece 2) for representing algorithms and tones .In this example of analysis is shown the first section of the whole composition only, which is represented entirely at the executive level, during the performance. Petri nets seems to be a very good model to analyse structural and thematic events of an electroacoustical composition.

It is more difficult to describe the differents levels in timbral parameters. In fact these can be associated to midi channel only .

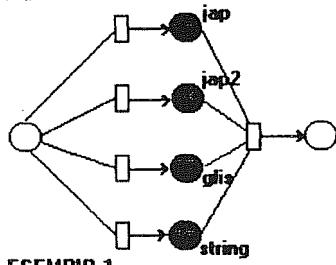
In questo articolo viene proposto un esempio di analisi e rappresentazione di un brano di musica elettroacustica mediante Reti di Petri. Questo genere musicale, com'è noto, non è facilmente rappresentabile su pentagramma e risulta quindi difficile analizzarne e visualizzarne gli eventi.[1] Le Reti di Petri sembrano offrire, almeno in parte la soluzione a questi problemi consentendo di creare partiture alternative. Gli eventi musicali con riferimento ai parametri di altezza, durata, intensità etc. sono associati ai posti, mentre le relazioni fra gli eventi stessi ed i loro rapporti strutturali dipendono dallo svolgimento e dalla struttura delle reti [2]. Si potrà quindi descrivere una composizione in rapporto agli elementi tematici, oppure individuarne le caratteristiche in relazione a parametri piu' complessi da

analizzare, quali il timbro. Il brano descritto in questo articolo è *Resonant Wholes*, di Claudio Gabbiani e Simonetta Sargent, interamente realizzato con la stazione di lavoro M.A.R.S. dell'Iris, gruppo Farfisa-Bontempi, una composizione per suoni di sintesi ed elaborazione elettronica dal vivo. Per poter mettere in relazione gli algoritmi con la rappresentazione mediante reti di Petri, l'intero brano è stato descritto a livello esecutivo con reti visualizzabili durante l'ascolto. Nell'articolo viene invece proposto un esempio di analisi relativo alla sola sezione iniziale del brano, della durata di un minuto e mezzo circa. Si è cercato di evidenziare in primo luogo lo svolgimento della composizione e le sue caratteristiche per così dire tematiche, ricorrendo anche alla possibilità di creare più livelli di astrazione fornita dalle reti. Si è poi passati alla descrizione degli algoritmi di sintesi basandosi sugli stessi principi.

La composizione, prevede, per la sezione analizzata in tutto quattro tracce, registrate su sequencer in modo da utilizzare dei files midi. La prima parte dell'analisi consiste nella descrizione dei vari livelli gerarchici del brano per rilevarne le caratteristiche strutturali e tematiche e ricostruire una partitura. La rappresentazione più generale, è data da una rete che sintetizza lo

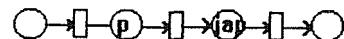
svolgimento di ciascuna traccia in posti che rimandano a diverse sottoreti (esempio 1)

Ciascuna traccia, può essere visualizzata partendo dal nodo macro corrispondente: dal posto chiamato *jap ch.2*, si ottiene la sottorete ad esso collegata, che comprende un unico oggetto musicale ulteriormente articolato a livello esecutivo in due posti: una pausa e l'evento sonoro *jap*: quest'ultima rete può essere eseguita e da luogo ad una partitura.



ESEMPIO 1

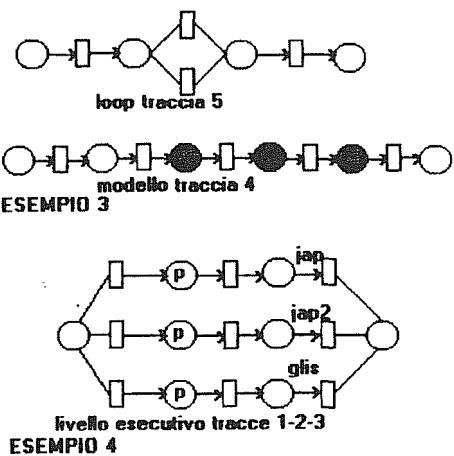
Lo stesso procedimento si può seguire per le altre tracce: il nodo macro *jap ch.3* della rete di livello più alto, richiama una sottorete costituita da un'unico oggetto musicale, comprendente a livello esecutivo una pausa ed un evento sonoro (esempio 2)



ESEMPIO 2

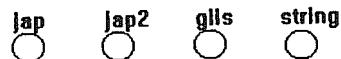
Si può notare a questo punto l'analogia delle tracce analizzate. Il nodo macro *gliss ch.5*

richiama una sottorete raffigurante la struttura loop in base alla quale l'oggetto musicale contenuto nella traccia, viene eseguito due volte. Anche in questo esempio gli oggetti musicali non subiscono trasformazioni, ma vengono solo eseguiti. Il nodo *string ch.4*, contiene un'altra rete di livello alto da cui si ottengono tre diverse esecuzioni del medesimo oggetto *string*, originale e trasformato (esempio 3).



Da questa prima descrizione ottenuta con un procedimento top-down, se ne è poi ricavata una seconda che tende ad evidenziare ulteriormente le caratteristiche già definite. Poichè l'oggetto *glis*, è costituito da un inciso A, autonomo e da un inciso B che riprende l'elemento tematico *string*, questa traccia è stata articolata in due differenti.

Anche qui si è scelto il metodo top-down, partendo dal livello più alto, per giungere al livello esecutivo. Si ottengono così due rappresentazioni a livello più generale e varie sottoreti particolareggiate (esempio 4). Le reti del livello esecutivo sono costruite in maniera deterministica per consentire la successione degli eventi del brano nell'ordine originario. Inoltre dal procedimento top-down adottato si ricavano anche gli incisi tematici definiti nei posti (esempio 5).

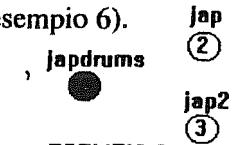


ESEMPIO 5

A questo punto dell'analisi si può affermare che le reti di Petri offrono uno strumento valido per individuare eventi musicali determinati da parametri di altezza, durata e dalle loro trasformazioni e si rivelano inoltre assai dutili, consentendo diverse rappresentazioni delle medesime strutture. Tuttavia la musica elettroacustica è principalmente caratterizzata dai parametri timbrici, si è voluto quindi realizzare una rappresentazione di questo aspetto, particolarmente significativo sempre mediante l'uso delle reti, partendo dal presupposto che un oggetto musicale può essere associato non solo a sequenze di note, ma anche a parametri di controllo; il

timbro puo' quindi essere definito semplicemente dal canale midi cui viene associato. La stazione di lavoro utilizzata per la composizione del brano qui analizzato, permette di creare algoritmi di sintesi ed elaborazione del suono, con l'ausilio di un programma grafico che visualizza i componenti dell'algoritmo e le loro relazioni, consentendone una comprensione adeguata senza una ulteriore descrizione mediante reti, peraltro possibile. E' invece interessante notare che in questi algoritmi sono presenti più livelli gerarchici: l'*algo* o struttura di base che rappresenta le connessioni esistenti tra i moduli per la sintesi o per l'elaborazione del segnale e i vari *tones* che sono un'articolazione ulteriore dei parametri dell'algoritmo stesso. L'*algo* corrisponde al livello piu' alto di rappresentazione con le reti di Petri ed il tone al livello di sottorete da eseguire. Una possibilità per rappresentare gli questi algoritmi con le reti di Petri sembra dunque essere quella di definire ciascun timbro come posto, senza attribuirgli altro parametro che il canale midi. Dato che ciascun algoritmo puo' dar luogo a piu' timbri diversi partendo dalla stessa struttura, questa differenza puo' essere definita in termini gerarchici costruendo posti che richiamano sottoreti.

Nella nostra composizione sono presenti quattro diversi *algo*; *Japdrums*, che comprende due *tones* è descritto in un nodo macro cui corrispondono i due oggetti sottorete *jap* e *jap2* definiti rispettivamente dal canale MIDI 2 e 3 (esempio 6).



ESEMPIO 6

Le reti di Petri costituiscono dunque un modello valido per la rappresentazione di strutture tematiche della musica elettroacustica, altrimenti di difficile descrizione; più problematica risulta invece una approfondita descrizione dei parametri timbrici associati ai canali midi.

References.

- [1] P.Schaeffer:"*Traité des objets musicaux*", Paris, 1966,1985
- D.Smalley:"*Spectro-morphology and structuring processes*",in S.Emmerson:"*The language of electro-acoustic music*",London, 1986
- S.Sargent:"*Per una definizione delle unità percettive di Aquatisme*", in "Secondo convegno di Analisi Musicale",Trento;1992
- [2]G.Haus,A.Sametti:
"*SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra*",IEEE Computer,vol.24,N.7,1991

THE COMPOSITIONAL PROCESS AND TECHNOLOGICAL TOOLS: AN APPRAISAL OF ALGORITHMIC COMPOSITION AS IT RELATES TO COMPOSITIONAL PROCESS

Dr. Noel Zahler, Co-director
Professor of Music
Center for Arts and Technology
Chair, Department of Music
P.O. Box 5632
Connecticut College
270 Mohegan Avenue
New London, Connecticut 06320
nbzah@mvax.cc.conncoll.edu
nbzah@conncoll.bitnet

Introduction

"The misunderstanding that lies in wait for us, and which we should mistrust terribly, is that of confusing composition and organization."

Pierre Boulez, "...Auprès et au loin"
[1, p.183]

The statement above, made in 1954, might be considered more prophetic today than we might have suspected at the time of its pronouncement. We are now inundated with computer assisted composition programs that generate output *ad infinitum*.. The results of these calculations must not be considered compositions. They are "raw data" from which composers may wish to draw compositional material. To interpret these calculations in any other manner is to misunderstand the very act of composition itself. When we are asked to accept this output as a "work of art" we

usually do so dependent on philosophic rather than aesthetic or artistic criteria. Such compositions are, more often than not, appreciated for their academic significance as they lack a sufficient complexity of relations to engage the listener more fully.

In fact, because of our interest in identifying formalisms to be included in computer assisted composition programs one might argue that we have become so fascinated with process, data manipulation for its own sake, that we may, at times, forget that these formalisms are a direct result of the perception of compositional details. It is the consequence of a compositional strategy which allows us, through reflection, to perceive a multifarious network of relations which is the composition. If we try to convince ourselves that we hear

relationships that we want to hear in a given composition because of an apriori assumption then we are doomed to repeat the mistakes of the past. For some, the engagement with and the purpose of, the compositional process seems to have become obscured entirely because of a reliance on the strategies of a computer assisted composition program.

As users of a powerful and growing ever more powerful technology we must ask questions about the need and use for the tools we build in light of the empirical and aesthetic demands we make on them. What are the advantages of these tools and what are our own needs for them?

This paper explores the ground upon which a number of well-known algorithmic composition programs have been based and scrutinizes both the procedures, as well as the results in search for a model by which we may reflect on the significance of the work of art and the means by which we create and perceive that work. Inquiry is made of a number of methodologies associated with algorithmic composition and we consider from an analytic (perceptual) viewpoint both the process and the result. The author's own compositional program (*Music Matrix*) is considered in light of the discussion. The appraisal suggests that it might be more

profitable to put our faith in the faculties of professional composers and our energies into specific types of programs that, in fact, "organize" and allow the micro and meta decision making process to take place using strategies other than the "automatic."

Computer Assisted Composition vs Automatic Composition

We must, from the start, distinguish between "computer assisted composition programs" and "automatic composition programs." The former seeks to help a composer with the various and laborious manipulation of data while the latter seeks to have the machine compose. We are not interested in machine composition in the present discussion. We are interested in the machine obviating details of a musical structure which the composer deems relevant to his/her conception of the composition.

Surveying the literature concerning computer music composition one is immediately struck by the confusion which seems to surround computer Assisted composition of any type. It is our

contention that computer assisted composition is a tool for the experienced composer. It is a means for exploring compositional material which has been invented by the user and helps in the discovery of relationships which may not be immediately obvious or which the composer hopes to employ as a *geheimnis* of the envisioned composition.

First, and foremost, is the composition. We define that composition as a mental construct found in the mind of the composer. It is the task of any given computer assisted composition program to help facilitate the composer's realization of that composition as distinct from the computers realization of some facet of the composer's idea. All too often computer realizations are one dimensional generalizations dependent on specific algorithms rather than on the composer's own vision or compositional imagination.

Algorithmic Composition

"An algorithm may describe a set of rules or give a sequence of

operations for accomplishing some task, or solving some problem [2]." In essence, with regard to music composition, an algorithm is a set of instructions which seeks to simulate a compositional formalism. Algorithmic composition has in many ways defied definition. We recognize the terrain, but in a similar manner to a "colloquialism" there is no generally agreed upon definition or list of techniques or methods for implementing such a program. The approaches to devising such formalisms are as numerous as those who attempt to write the programs. Perhaps this is how it should be, for the compositional process, in terms of a methodology, is entirely subjective. By its very nature it eludes formal definition. So what is it that we hope to capture when writing an algorithmic computer assisted composition program? What part of the compositional process can we possibly hope to capture? In our opinion the answer to both questions lies in dealing with the most formal procedures of music composition in as straight

forward and intuitive a manner as possible. This is no simple task, for as Gareth Loy argues "[t]he creation - and sometimes even the appreciation - of formalisms generally takes an exertion of great mental effort because one must be holding the goal of the method and the requisite actions in mind, while simultaneously observing and recording what one is doing to achieve the goal." But, while extolling the virtues of formalisms Loy quickly points out that "most formal procedures used in music are not strictly algorithmic [2]." If the latter statement is correct then it must follow that any attempt to create an "algorithmic composition program" is contrary to the very nature of the beast.

The Music Matrix

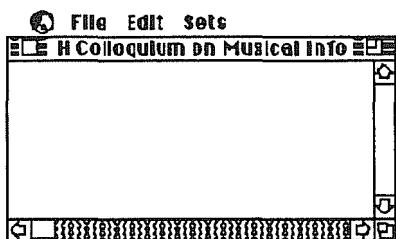
The *Music Matrix* is an original composition tool created by this writer and Jonathan Kozzi. Although it is still in its development stage it offers a number of advantages to composers and theorists studying the realm of pitch relations. The program is not intended to compose. It is simply an aid to the experienced composer

needing to explore pitch material he/she has already invented. The *Music Matrix* organizes pitch according to very specific instructions from the composer. The *Music Matrix* is a Macintosh program written in the C language. It is a composer's/analyst's tool for defining pitch structures. *Music Matrix* will generate lists of pitch transformations for user defined sets of any ordering or cardinality. It will list transpositions, inversions, retrogrades, retrograde inversions, M5 and M7 operations, as well as other user defined transformations. It will decline and define subsets, list interval vectors, Z related pairs, and build combinatorial, as well as user defined matrices. The program is equipped so that users may specify the types of transformations to be used in building matrices. When directed by the user the program will build aggregate or weighted pitch structures. All information compiled by *Music Matrix* can be saved, printed or transferred to any word processing program for further formatting. In addition, all output is midi compatible and can be

routed to external synthesis equipment to be heard.

Music Matrix is a typical Macintosh program using pull down menus and dialogue boxes to communicate with its user. When launched, three menus and a blank window are presented (ex. 1). The File and Edit menus, respectively, allow the user the usual Macintosh options

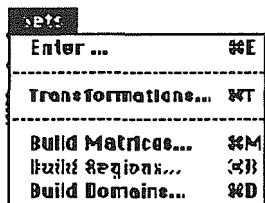
Ex. 1.



for opening, closing, printing, saving, and allocating preferences; cut, paste, and copy functions, etc. It is the third menu, Sets, (ex. 2) which initiates the most important features of the program.

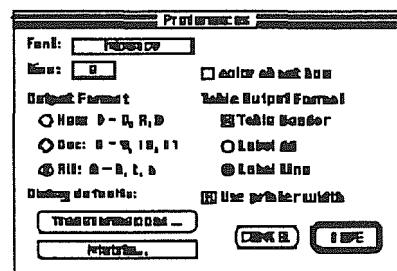
Pitch is referenced throughout the program

Ex. 2



through integer notation with the numbers 0-11 representing the chromatic scale modulo 12. When accessing the "Preferences" level of the File menu a dialogue box is activated (ex. 3)

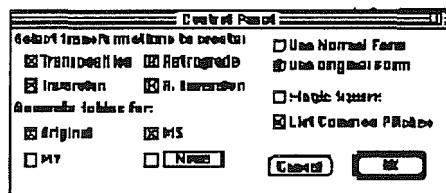
Ex. 3



which gives the user the choice of specifying pc's (pitch classes) using hexadecimal notation, decimal notation, or a generally accepted alternate notations using the integers 0-9 and the letters t (ten) and e (eleven) to eliminate any confusion in base 10 notation. Once the form of input notation is selected a second dialogue box (the "Control Panel") is accessed (ex. 4.) which allows the user to select the types of lists they would like to have declined when selecting the Transformations level of the Sets menu in screen 1 (each category of

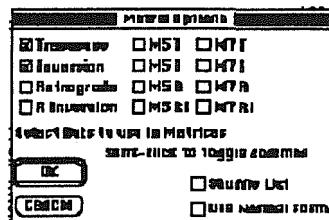
transformation appears in it own screen).

Ex. 4



These include a choice of all forms of the entered set listed according to "normal form" or the original form which was input by the user. If the "set" was a twelve tone set there is an option for a "magic square." Traditional serial transformations such as transposition, retrogression, inversion and retrograde inversion are all available. In addition, tables can be generated for M5, M7, and any user defined multiplicative operation deemed necessary. Common pitches between the original pc set and its transformations can be listed and combinatorial matrices will be generated for collections possessing those qualities. The latter function calls up yet another dialogue box (ex. 5) which allows the user to choose which of the already derived

transformations will be used in generating the matrices. Two other options are available, "Shuffle List" and "Use Ex. 5.

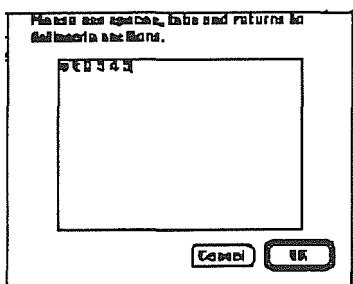


Normal Form." The "Shuffle List" command randomizes the order of matrices and the "Use Normal Form" allows the user to disregard the ordering of the original set and generate all matrices using the normal form of the collection.

Having navigated through these last three dialogue boxes we have set all the preference parameter for a particular look at a pitch class set. In referencing the Input selection of the Sets menu we access the input screen (ex. 6). This window allows entry of up to twenty-five (25)characters horizontally and fifteen (15) vertically. A single line or matrix of the users choice may be entered allowing the exploration of harmonic or melodic elements of the material.

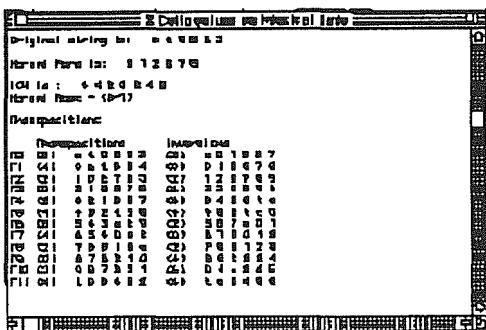
The first data screen,(ex. 7), displays the original string (set) at the top of the screen followed by its normal form, interval

Ex. 6.



class vector, and prime form name (from the

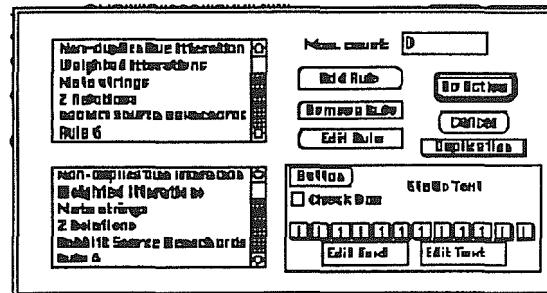
Ex. 7.



Morris Listing). If the set has a "Z relation," the prime form, name, and normal form of that related set is displayed as well. Below this are the lists of transformations previously tailored in the "Control Panel." By accessing the "Set" menu

again and clicking on the "Build Matrices" item another dialogue box appears (ex. 8). This panel is at the heart of the program

Ex. 8.



effectiveness. Here the user may use any of the stock rules to operate on their material or create their own rules to search and find transformations of the collection they believe might be useful. The rules already incorporated into the program appear in the boxes on the left side of the screen. The various other buttons allow the user to customize their selection.

The Max Count window allows the user to specify the number of iterations of a particular transformation. This is useful in situations where there is limited memory. The box in the lower right hand side of the screen allows the user to weight collections by changing the

index in each of the small boxes. The twelve boxes represent each of the possible twelve pitches found in the user's collection. The Do button allows the user to apply the Ex. 9.

H Collection Hexachord											
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113

rules selected and move to the next data window (ex. 9). Returning to the Set menu again and accessing the Build Regions and Build Domains levels (examples 10 and 11) brings about similar actions organizing transformations of this hexachord into full twelve-tone sets and 12X12 matrices of this set.

While the above discussion has concentrated on twelve tone applications, the authors want to emphasize that a much broader range of applications are possible. In addition to refining the present program we are developing a program to

transform time dimensions in a similar way.

Ex. 10.

H Collection Hexachord											
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113
211111	211112	111113	211113	111113	111113	111113	111113	111113	111113	111113	111113
111111	111112	111113	111113	111113	111113	111113	111113	111113	111113	111113	111113

Ex. 11.

H Collection Hexachord											
011111	221111	111111	111111	111111	111111	111111	111111	111111	111111	111111	111111
111111	011111	221111	111111	111111	111111	111111	111111	111111	111111	111111	111111
211111	221111	011111	111111	111111	111111	111111	111111	111111	111111	111111	111111
111111	221111	211111	011111	111111	111111	111111	111111	111111	111111	111111	111111
211111	221111	211111	211111	011111	111111	111111	111111	111111	111111	111111	111111
111111	221111	211111	211111	211111	011111	111111	111111	111111	111111	111111	111111
211111	221111	211111	211111	211111	211111	011111	111111	111111	111111	111111	111111
111111	221111	211111	211111	211111	211111	211111	011111	111111	111111	111111	111111
211111	221111	211111	211111	211111	211111	211111	211111	011111	111111	111111	111111
111111	221111	211111	211111	211111	211111	211111	211111	211111	011111	111111	111111

Conclusion

As our tools become ever more powerful and we are tempted by the proverbial "automaton" we must be careful not to destroy the very essence of the act of music composition. We must take special care to allow the proper amount of interaction between each individual and the machine. Again, Boulez comments and we who create these tools should take heed:

"let us rather see the work as a series of refusals in the midst of so many possibilities; one must make a choice

and it is there that we encounter the difficulty so well side stepped by the expressed desire for 'objectivity.' Such choosing is precisely what constitutes the work, being renewed at each instant of composing; the act of composition will never be identical with the juxtaposition of the confrontations established in an immense statistic."

Pierre Boulez, "...Auprès et au loin"
[1, p. 204]

Acknowledgments

Gratitude to Analysis and Technology, Inc. for financial support of this project and to the Center for Arts and Technology at Connecticut College for continuing support.

Selected References

- [1]Boulez, Pierre. *Notes of an Apprenticeship*, Alfred A. Knopf, New York (1968).
- [2]Loy, Gareth, "Composing with computers," in *Current directions in computer music research*, ed. Max Mathews and John Pierce, The MIT Press, Cambridge, Massachusetts(1989).
- [3]Bain Reginald. "A Musico-linguistic Composition Environment." *Proceedings*, ICMC 1988, San Francisco, p.102.
- [4]Baisnée, Pierrre-Francois, et al. "Esquisse:A Compositional Environment." *Proceedings*, ICMC 1988, San Francisco, p108.

[5]Morris, Robert D. *Composition with Pitch-Classes*, Yale University Press, New Haven and London (1987).

[6]Roads, Curtis, ed. *The Music Machine*, The MIT Press, Cambridge, Massachusetts (1989).

[7]Roads, Curtis, and John Strawn, eds. *Foundations of Computer Music*, The MIT Press, Cambridge, Massachusetts (1989).

[5]Cope, David. *Computers and Musical Style*, A-R Editions, Inc., Madison (1991).

[6]Ames, Charles. "Automated Composition in Retrospect," *Leonardo* 20(2) (1987).

[7]Knuth, Donald E. *The Art of Computer Programming, Volume 1-Fundamenta Algorithms*, Addison-Wesley, Reading, Massachusetts (1973).

[8]Baird, Bridget, Donald Blevins, and Noel Zahler. "Implementing an Interactive Computer Performer." *Computer Music Journal*, Vol. 17, N.2, pp.73-79, 1993.

Capitolo 3

PERCEZIONE

THE PERCEPTION OF MUSICAL STRUCTURE AND TIME

David R. Keane, Lola L. Cuddy,
Carole A. Lunney, and Jennifer Dufton

School of Music and Department of Psychology
Queen's University, Kingston, Ontario, Canada K7L 3N6
FAX 1-613-545-6808
E-mail keane@QUCDN.QueensU.Ca

Introduction.

The experimental research reported in this presentation was part of a project concerned with the perception of musical time within large-scale structures. Specifically, we were interested in quantifying the perceived passage of time and its relationship to the formal structure of a musical piece. A musical piece was composed especially for the experiment. From a research aspect, this strategy had a number of advantages: it ensured that, within a musically valid stimulus context, a number of methodological constraints on the experimental techniques could nevertheless be honoured, and it ensured that the piece was equally unfamiliar to all listeners tested in the experiment. Moreover, it provided an opportunity to compare experimental findings with the composer's intentions and intuitions, thus yielding hypotheses for further investigation.

The objective, from the composer's point of view, was to make a piece that would be engaging and listenable, using 20th-century harmonic and melodic features. Although the piece was written specifically for the experiment, the aesthetic approach was virtually the same as the composer would use in any musical composition. The one difference in technical approach is that the structure was derived primarily from textural, rhythmic, dynamic and melodic processes. Typically, the composer would emphasize timbre in equal measure with the foregoing parameters, but he decided in this instance, for the purpose of simplification, to limit the timbral range to that comparable to a conventional acoustic instrument.

The musical piece.

The composer was asked to provide a short (4-min) poly-

phonic composition. Conventional (Western tonal) functional harmony was to be avoided, but the piece was to contain a formal, metric, and motivic structure that could be apprehended by our sample of listeners. A requirement of the experimental technique was that the piece contain no literal repetition of phrases, or repeats. The composer provided a detailed segmentation analysis of the piece, but was not told anything about the specific motivation for the experiments.

The composer's description follows: he set out to create a work with some reference to tonal centres, but with distinctly nonfunctional relationships among pitches. The resulting music is rhythmically and metrically fairly conventional, but, in the main, the harmonies and melodic line avoid triadic or other formulaic reference. The work draws quite freely upon the 12 elements of the chromatic scale. It has a clear sense of phrase and of cadence, but there is not-infrequent elision of phrases. The first (30 bars) of the three sections treats several thematic/textural ideas. The second section (57 bars) develops these melodically and harmonically and the third (31 bars) recapitulates the first section, but the materials are somewhat elaborated and the segmentation

into phrases is less simple and clear cut. The work is structured largely through progressive (especially dynamic, harmonic, and melodic) strategies, harmonic/melodic sequences, and cadential breaks. The general flow of the piece, however, could best be described as capricious.

Figure 1 displays the first two notated lines of the piece. The piece was composed using Notator 3.1 software on an Atari 1040ST computer. The MIDI file for the piece was then transferred to a Zenith Z-200 computer, which controlled both the presentation of the piece to listeners and experimental tests based on the piece. Cakewalk Professional Software 4.0 was used to create the test files used in the experiment. The piece and the experimental tests were realized with the pick-guitar setting of a Yamaha TX-802 synthesizer.

Experimental procedures.

For one component of the experiment, the methodology was adapted from a study by Clarke and Krumhansl [1] in which listeners were asked to identify the location and duration of short segments extracted from musical pieces. These authors proposed that



Figure 1. The first eight bars of the piece.

the pattern of listeners' judgments reflected their sense of the rate of musical development over time. The present experiment examines and expands this proposal.

A second component of the experiment collected listeners' impressions of the piece. This aspect of the experiment was of particular interest to the composer because it offered systematic feedback of listeners' impressions as the listeners became acquainted with the composition. Rarely does a composer have much more than his/her intuition as a guide to such decisions as amount of repetition, degree of digression, length of sections, etc. Although having responses of the kind generated by this study does not relieve composers of difficult decisions, the responses do offer some support for strategies that previously had no more frame of

reference than a composer's hunch.

Sixteen musically trained listeners (6 males and 10 females, aged 19 to 22) participated. They heard the entire musical piece 20 times--4 sessions of 4 listenings each, plus 4 additional listenings on the testing date. After each session, listeners completed a two-part questionnaire.

The first part asked the subjects to rate the piece on 10 subjective dimensions, such as "active vs. inactive" and "pleasant vs. unpleasant". The second part contained five questions addressing listeners' impressions of specific features of the piece (i.e., rhythm, melody, motifs, pitch structure).

On the testing date, after hearing the piece four times, the listener heard a series of 22

segments abstracted from the piece. The entire piece was represented across the segments. The segments varied in duration from 11.5 to 25 sec, and were presented in random order. Listeners were asked to estimate the location and duration of each segment on a horizontal time-line, which represented the total duration of the piece (1 mm = 1 sec). After hearing each segment, listeners estimated where it started and ended by marking two vertical lines on the time-line. A separate time-line was provided for each segment. After all 22 judgments were completed, the segments were played again, and listeners were asked to rate the complexity and completeness of each segment. The listener was asked to consider complexity as the perceived amount of musical information contained in the segment, and completeness as the perceived form of the musical phrase--similar to that of the perceived form of complete and incomplete sentences.

Results from the first component of the experiment.

For each segment, for each listener, the midpoint between the two vertical lines drawn on the time-line was measured in mm and it served as the estimate of perceived location of the segment. A correlation was

computed, for each listener, between perceived location and actual location of the segments. (A high correlation is evidence of a linear relation between estimates of musical time and real time.)

For half the listeners, the correlation was statistically significant ($p < .05$). The average data for these listeners, called the High correlation group, is shown in Figure 2. Also in Figure 2 is the average data for the remaining listeners, called the Low correlation group.

The functions shown in Figure 2 were obtained by applying a resistant smoothing technique to the obtained data. There is a nonlinear trend in both functions, which is much more apparent in the Low group. Both groups display a monotonic rise in the function up to a point toward the end of the middle section of the piece (real time about 150 sec, bars 74-75). The slope is much steeper for the High than for the Low group. After that, the function flattens for the High group and actually reverses for the Low group. At the end of the piece, for both groups, the last segment was clearly distinctive and was located at the far end of the time-line. (Note: The mean for the final

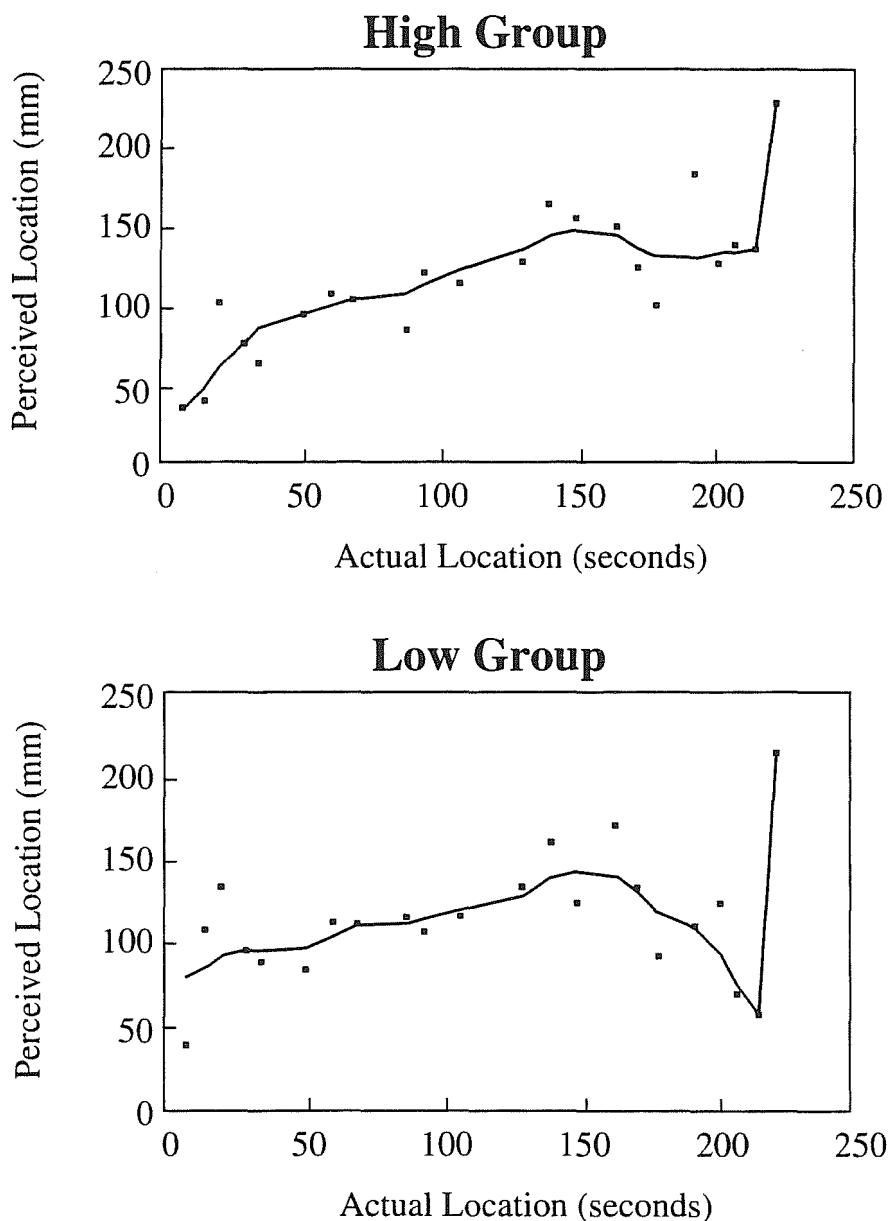


Figure 2. Perceived location of segments as a function of actual locations in the piece.

segment was not included in the smoothing procedure, to avoid undue influence from this point.)

What is responsible for the nonlinearity in the estimates of perceived location? A possible answer to this question lies in the listeners' ratings of complexity and completeness. As the piece progressed, there were gradual shifts in both these ratings for both groups of listeners. Perceived complexity tended to increase during the development section and remained high, on the average, until the end of the piece. Perceived completeness (with the exception of the final segment which was judged "very complete") tended gradually to fall from the end of the development to the penultimate segment of the piece. There were, of course, differences in ratings from segment to segment, many of which corresponded to the composer's segmentation analysis. It is the general trends, however, that are instructive on the question above. These general trends reveal perceptual differences between the first and third section of the piece, differences which converge with the composer's description of the greater elaboration and less simple segmentation of the third section.

The results shown in Figure 2 suggest that, as the piece pro-

gressed, there was a tendency to overestimate the amount of real time remaining until the end of the piece. The trends in the complexity and completeness ratings suggest that this extension of time was accompanied by a sense of increased musical information, and a sense of increasing incompleteness--likely resulting in increasing musical tension. Musical time "slows down" to accommodate this increasingly rich musical experience.

Results of the second component of the experiment.

Eight of the 10 rating scales yielded judgments that differed statistically from the neutral point of the scale. The listeners' responses showed that they found the piece "active, restless, unconventional, structured, interesting, and playful". Ratings on two scales shifted during the listening sessions. The piece was judged as becoming more "predictable" and more "pleasant". If we assume that the piece was initially difficult to understand on the listeners' part, the shift in perceived predictability and pleasingness with stimulus repetition is supportive of Berlyne's theoretical approach to experimental aesthetics [2] [3]. However, it will be necessary to study musical

pieces of differing structural content to evaluate the contribution of this finding fully.

Concluding remarks.

We have shown that the methodology proposed by Clarke and Krumhansl [1] is a promising approach to studying listeners' sense of musical time. Our conclusions are necessarily speculative: we need next to know to what extent the experimental technique can identify sensitivity to different styles and structures, and to what extent the listeners' sense of musical time can be manipulated by varying properties of musical structure.

The composer finds the initial results supportive for his operating premise that, while contemporary works must evidence novel or innovative techniques (which may be initially distracting or alienating for the listener), such conventional musical features as cadential divisions and progressive patterns can be used to reduce bafflement, to engage listeners' attention, and to offer musical pleasure. At the same time, the degree of accuracy with which listeners are able to identify segments of the work in time and the generally more favourable response to the music upon increasing familiarity with it speak well for the average

listener's ability to enjoy and understand contemporary music. Of course, it is now important to vary structural strategies in similar studies to determine what effects each variable has on the listeners' responses.

[Experimental work was supported by a research grant from the Natural Sciences and Engineering Council of Canada to L. L. Cuddy. D. R. Keane is the composer of the piece.]

References.

- [1] E. Clarke, C.L. Krumhansl: "Perceiving musical time", *Music Perception*, Vol.7, N.3, pp. 213-252, 1990.
- [2] D.E. Berlyne: *Aesthetics and psychobiology*, New York: Appleton-Century-Crofts, 1971.
- [3] K.C. Smith, L.L. Cuddy: "The pleasingness of melodic sequences: contrasting effects of repetition and rule-familiarity", *Psychology of Music*, Vol. 14, N.1, pp. 17-32, 1986.

TONE CENTER ATTRACTION DYNAMICS AN APPROACH TO SCHEMA-BASED TONE CENTER RECOGNITION OF MUSICAL SIGNALS.

Marc Leman

University of Ghent

Institute for Psychoacoustics and Electronic Music

Blandijnberg 2

B-9000 GHENT

Belgium

Email: ml@flw.rug.ac.be

1 Introduction

In this paper we focus on the problem of tone center recognition and interpretation. The model is part of a theory of musical morphology whose aim is to provide an operational account of music cognition in terms of psychoacoustics and dynamic systems theory. The model forms a basis for non-symbolic research in music imagination and has applications for music analysis as well as interactive music making.

Computer simulations, based on ear models and principles of self-organization, show that networks of artificial neurons, exposed to music, can develop functional organizations that are relevant for tone center perception [8] [9] [10] [11] [12] [13]. The network, trained with music, develops a response structure divided into ordered areas that reflect the circle of fifths for chords and tone centers. The ordering, as well as the response to chords and tone centers can be considered the

most important aspect of its emerging behavior. The results, which show a good agreement with the mental structures for tone center perception known from psychological investigations [7] have led to a theory of tone semantics [10] [11] [12].

The computer model shows that a schema for tone center perception emerges automatically - without effort. The schema is produced by a self-organizing neural network of the Kohonen-type [6] and the inputs are patterns than come from an ear model. No information form higher levels or ``domain specific knowledge'' is needed to explain the the emergence of the response structure, so the approach is purely *data-driven*. But in tone perception there is more involved than just long term learning. Mechanisms at the periphery of the auditory system, are continuously processing new information into auditory images. While these short term images may contribute to learning at long

term, their relevance as isolated auditory "objects" is far from evident with respect to intelligent action and decisions in an environment. To make any sense at short term, we must suppose that the information flow is somehow related to information that is stored in a long term memory. It can be argued that organisms equipped with such mechanisms have a high survival value. The basic principle of recognition and interpretation is that rapid changing patterns (that come from the senses) are related to memory structures (schemata) that are less vulnerable to such rapid changes. New information is resolved by the existing knowledge frame such that the organism can efficiently react to the stimuli of the environment.

The aim of this paper is to explore aspects of schema-driven perception in a tone center perception task. Although schema-driven perception is not well understood, [2] we present a framework of how stable structures of tone perception, once established, might be used in recognition and interpretation of musical signals. This involves a short term and *schema driven* interaction because it relies on organised information in memory.

2 A Model of Tone Center Perception

The computer model consists of two modules: a perception

module and a cognition module. The perception module is based on a psychoacoustical model of the ear [14]. The function of the ear model is to extract the relevant pitch information that is present in the signal. For tone center recognition we estimate that the relevant pitch information is to be found in the residue region, that is, between about 50 and 500 Hz [15]. The frequencies of this region convey the tone information of the music at a particular point in time. This information is represented by a so-called "tone residue image".

The ear model consists of two parts: an analytical part and a synthetical part. The analytical part, adapted from Van Immerseel and Martens, is based on a model of hair cells implemented as a bank of asymmetric bandpass filters at distances of one bark (one critical band). 20 such filters are used in the range of 220 to 7075 Hz (center frequencies). The ear model operates at a sampling rate of 20000 sa/sec, but the extraction of envelope patterns allows a down-sampling to 2500 sa/sec. The filters perform a non-linear frequency analysis of the musical signal. The output is a vector of 20 elements, each element representing the probability of neural firing within an interval of 0.4 msec.

The synthetic part of the ear model comprises a periodicity analysis in each channel and a

summation over all channels. Periodicity analysis is carried out by autocorrelation of the firing patterns. To sharpen the peaks, we clip the firing values to half of the mean value of the analysed frame. The autocorrelations are computed at intervals of 10 msec.

To account for the temporal dependencies between patterns, the tone residue images are integrated within a window of 3 seconds. The resulting images are called "tone context images" [13].

To obtain the tone center patterns, we calculate the last tone context pattern of 72 cadences to wit: all cadences of degree I-IV-V7-I, I-II-V7-I and I-VI-V7-I for major and minor, as well as their transpositions over the chromatic scale ($3 * 2 * 12 = 72$). Given these 72 tone context patterns, we take the mean of the three corresponding cadences for each tone center, which reduces our data to 24 ($72 / 3 = 24$) patterns. The relationships between the patterns show a similarity of 0.98 and 0.96 with the relationships in the data of Krumhansl.

3 The Metaphor

The tone center attraction dynamics operates as short term and relates tone context information to the schema. In particular, recognition is associated with the localization of short-term auditory images in

a space of tone center images, while interpretation is associated with an adaptive dynamics of the position of the short-term image within this schema. While we define recognition as a purely data-driven feature, it is assumed that interpretation involves the reconsideration of past interpretations in view of new contextual evidences.

The tone center attractor dynamics (TCAD) is based on a model of brain dynamics. It describes the behavior of the brain in terms of attractors, stable states and transitions towards stable states [5] [1]. The states that attract other, nearby, states are called attractor states. We assume that tone center images are stable states that serve as attractors for other, less stable, states - such as the tone context images. An image corresponds to the activity of neurons and is represented by an ordered array of numbers - a vector of pattern. A state is an image at a certain point in time. Two features of this dynamics require particular attention. First of all, in tone center perception, it turns out that recognition and interpretation is often ambiguous - in the sense that the perceived key is related to different tone centers at the same time. The attractor states are indeed highly related to each other. In Jazz music, for example, tone context patterns often have no pronounced tone center and part of the art is just to avoid the

attraction of tone centers. The tone context is then localized somewhere in the middle of different attractor states, and the forces of attraction have a relatively weak influence on the position of the percept - which gives more freedom to the performer. In addition, a feedback strategy may be necessary in order to adapt previous interpretations in the light of new evidence. For example, in the sequence IV-V-I, the first chord points to the tone center of the tonic of the chord and it is only after hearing the rest of the sequence that the interpretation of the first chord can be given in terms of a sub-dominant and dominant function to the tonic of I.

a Recognition

To simulate recognition we restrict the model to a determination of the position of the tone context image in a framework of tone center images. In other words, we just follow the tone context as it is produced by the data-driven processes, and compute the distance of the pattern to all tone centers in terms of similarity relationships. Examples of such an analysis are given in Leman [12].

b Interpretation

The approach described above is limited to a registration of the position of the tone context in a framework of tone centers. It

does not involve any activity of the schema. Therefore, problems such as those related to the interpretation of the IVth degree, are not resolved.

The metaphor of TCAD is that of an *elastic snail-like* object moving in the state space. The head follows the time index of the information stream and the tail corresponds to the P-states. The position of the object P is given by an *interpretation state* whose dimensionality is equal to the number of stable states. Its vector contains the distances of P to the tone center images and thus determines the position of P in the framework of stable points. The path followed by the head of our elastic *snail-like* object corresponds with the recognition process described in the previous paragraph. The tail, however, corresponds to a frame that keeps track of the past P-states. Thus, it consists of an array of interpretation-vectors, one for each P-state in the memory frame. In the current implementation, the determination of the position of the head is important because the interpretation of past the tail depend on it. And by changing from one attractor to the other, the interpretations will change too. The position of the tail is then reconsidered in the light of the new position of the head. But because the tail is susceptible to the forces of attractors, it may happen that the positions of parts

of the tail remain near one attractor, while the head and another part of the tail are near another attractor. That is what is meant by "elasticity": a demarcation of past interpretation states that is caused by the forces of different attractors.

4 Dynamics

In what follows we simulate aspects of the attractor dynamics by a computational method. The aim is basically to clarify our intuitions in these matters. The law of adaptation is defined as:

$$\begin{aligned}
 P(t, \tau)_i &= \\
 P(t-1, \tau-1)_i + & \\
 \alpha * \sum_{k \in A(t, 0)} &\left[\text{cor}\left[P(t, 0)_i, T^k_i\right] * T^k_i \right] + \\
 \beta * \sum_{k \in A(t-1, \tau-1)} &\left[\text{cor}\left[P(t-1, \tau-1)_i, T^k_i\right] * T^k_i \right] \quad (1)
 \end{aligned}$$

where i is an element of the vector P and α and β are scaling parameters. The summations run over all tone center images T that are within a distance (defined by a threshold) from P . These tone center images belong to the attractor-set A . The double time index is used to specify the absolute time (t) and the offset towards the past (τ). The equation says that the adaptation of $P(t, \tau)$ is based on the proper history, $P(t-1, \tau-1)$ plus a change based on the contribution of $P(t, 0)$ -attractors

and $P(t-1, \tau-1)$ -attractors. $\text{Cor}(\cdot)$ is the correlation function.

In general, the adaptation depends on the correlation between P and T . When the correlation is above the threshold (it satisfies the condition of the A -set), then P is in the attractor-basin of T and will be adapted such that it shifts a little bit more to T . There might be more than one T in the A -set. Obviously, Eq.1 applies only to a limited history of P -states. Therefore, the range of τ is limited to a few seconds. $P(t, 0)$ by itself is not adapted. To allow a better concurrence between the members of the A -sets we have rescaled the output $\text{cor}()$ such that all values above or equal to the threshold are rescaled to values between 1 and 0.

5 TCAD at work

The test example is the chord sequence FM-FM-CM-Cm-Gm-Dx(5)7-Gm-Gm-BbM-BbM-FM-Am(7)9-FM-FM. The expression " $x(5)7$ " means dominant chord minus the fifth and " $m(7)9$ " means minor chord with none minus the seventh. The sound was generated by a Csound program, using Shepard-tones with a duration of 0.5 sec per chord and a short attack and decay (to avoid clicking). For practical reasons we have summarised the results on a tone center map (Figure 1). The map shows the trajectory of the purely data-driven method ("recognition") in full line and

the schema-driven method ("interpretation") by the dotted line. For the latter, the position of the states was taken after a two seconds of adaptation. The trajectory of the interpretation compensates the time-delay that is introduced by the integration of the tone residue patterns (not shown on the figure). Also the path is slightly different. In general, the TCAD-analysis seems to give a more accurate estimate of the tone context.

6 An approach to psychoacoustic-based harmonic analysis

a *An approach to psychoacoustic-based harmonic analysis*

A TCAD-analysis outputs the position of a particular tone context image with reference to a frame of tone center images. Apart from its interest as a "fine-grained" tonal analysis, it is to be expected that this information can be of help in defining functional roles of tones and chords.

We are currently investigating this application in the framework of HARP (Hybrid Action Representation and Planning) [3] [4].

b *Tone center information in interactive systems*

In interactive music systems, by which we understand systems whose real-time sound

generation is based on a sound environment that includes its own acoustical signals, we believe that the model of tone center recognition might provide important information for the navigation in the tone center space. Indeed, it is not unrealistic to suppose that very soon, the recognition part of our model can be realised in real-time. Much more computational power would be needed, however, to cope with the interpretational part of the model.

7 Conclusion

We presented a framework for schema-based tone center recognition and interpretation. Recognition was associated with the localization of short-term auditory images within a schema of tone center images. Interpretation was associated with an adaptive dynamics of the position of the percept within this schema. The model assumes that interpretation involves the reconsideration of past interpretations in view of new contextual evidences.

Acknowledgements

We wish to thank H. Sabbe and the Belgian National Science Foundation (F.K.F.O) for support, as well as N. Cufaro Petroni for discussion during a stay in Ghent. A. Camurri, J. P. Martens, L. Van Immerseel and B. Willems are greatful for their help. The scientific responsibility is assumed by the author.

References

- [1] D.J. Amit, "*Modeling brain function: the world of attractor neural networks*", Cambridge University Press, Cambridge, MA, 1989.
- [2] A.S. Bregman, "*Auditory scene analysis: the perceptual organization of sound*", The MIT Press, London, 1990.
- [3] A. Camurri, C. Canepa, M. Frixione, R. Zaccaria, "*HARP: A framework and a system for intelligent composer's assistance*" In D. Baggi ed., Readings in computer generated music, IEEE Computer Society Press, Los Almitos, CA, 1992.
- [4] A. Camurri, C. Innocenti, C. Massucco, R. Zaccaria, "*A software architecture for sound and music processing*", 35 (1-5), *Microprocessing and Microprogramming* (Proc. EUROMICRO-92), Paris, 1992.
- [5] H. Haken, M. Stadler, "*Synergetics of cognition*", Springer-Verlag, Berlin, 1990.
- [6] T. Kohonen, "*Self-organization and associative memory*", Springer-Verlag, Berlin, 1984.
- [7] C.L. Krumhansl, "*Cognitive foundations of musical pitch*", Oxford University Press, New York, 1990.
- [8] M. Leman, "*Symbolic and subsymbolic information processing in models of musical communication and cognition*", 18(1-2), 141-160, *Interface - Journal of New Music Research*, 1989.
- [9] M. Leman, "*Emergent properties of tonality functions by self-organization*", 19(2-3), 85-106, *Interface - Journal of New Music Research*, 1990.
- [10] M. Leman, "*Een model van toonsemantiek: naar een theorie en discipline van de muzikale verbeelding*", Doctoral Dissertation, University of Ghent, Ghent, 1991a.
- [11] M. Leman, "*The ontogenesis of tonal semantics : results of a computer study*", In P. Todd & G. Loy (eds.), *Music and connectionism*, The MIT Press, Cambridge, MA, 1991b.
- [12] M. Leman, "*The theory of tone semantics: concept, foundation, and application*", (2), 345-363, *Minds and Machines*, 1992a.
- [13] M. Leman, "*Tone context by pattern-integration over time*", In D. Baggi ed., Readings in computer generated music, IEEE Computer Society Press, Los Almitos, CA, 1992b.

[14] L. Van Immerseel, J. P. Martens, "Pitch and voiced/unvoiced determination with an auditory model", 91(6), 3511-3526, The Journal of the Acoustical Society of America, 1992.

[15] E. Zwicker, H. Fastl, "Psychoacoustics : facts and models", Springer-Verlag, Berlin, 1990.

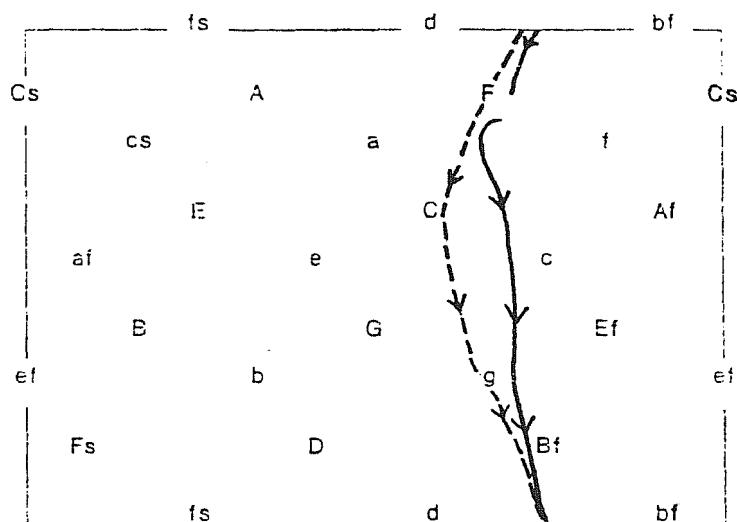


Figure 1 Trajectories of recognition and interpretation on a tone center map. The figure is a torus in that left and right sides as well as upper and lower sides are connected.

Capitolo 4

RETI NEURALI

AUTOMATIC PERFORMANCE OF MUSICAL SCORES BY MEANS OF NEURAL NETWORKS: EVALUATION WITH LISTENING TESTS.

G. U. Battel*, R. Bresin[†], G. De Poli[†], A. Vidolin*

*Conservatorio di Venezia "Benedetto Marcello"

[†]D.E.I. - C.S.C. Università degli Studi di Padova
via Gradenigo 6/a - 35129 Padova - Italy
phone: +39 49 8287631- fax: +39 49 8287699
E-mail: depoli@dei.unipd.it, rb@csc.unipd.it

Introduction

Musicians, according to the instrument they play, make loudness, duration and timbre deviations on the notes of the score they are performing, since the traditional musical notation does not suffice the composer's real intentions, and leaves some freedom's degrees to the player himself. These deviations determine the performing characteristics of a pianist in respect to another one. Furthermore a literal performance of a musical score would lead to an extremely mechanic and unnatural performance to the listener's ears.

The present work starts from the Sundberg's and co-workers' researches on automatic scores' performance [1] [2] and continues the research on real-time piano scores performance by

mean of particular artificial neural networks. In our previous works [3] [4] we showed the possibility to build some neural networks which can learn some performing rules. These nets show good generalization properties, and, after a training phase, are able to do real-time performances of any score introducing some appropriate deviations.

In the present research we propose a comparison test between various performances to evaluate, by mean of listening tests, the use of trained neural nets in automatic performance.

Description of the experiment

Methods

Materials

Two tonal melodies were chosen to be used in this study.

The first one is the theme of the third tempo of the Mozart piano sonata K. 284, the second is the theme of the first tempo of the Mozart piano sonata K. 331.

For the experiment it has been used a subset of Sundberg's rules: the selected rules influence mainly the relations between near notes and don't involve greater segments. In this way the selected rules are more suitable to a structurally simple musical example and to a strictly classical repertoire. This formal need, and since the sonata K. 331 was previously used in other works [1], brought us to the choice of the two themes of Mozart. The starting hypothesis are two: it is possible to obtain an acceptable performance with small performing deviations and without involving the great form; for the intrinsic characteristics of Mozart's music, the melodies are performable in a meaningful way on their own, and the deviations sound pleasant and understandable to the listener.

We performed each of the two melodies in three ways: deadpan (with no expression); with expression given by a subset of Sundberg's performance rules, in the following these melodies will be called rules-melodies; with expression given by two neural networks trained with the same subset of Sundberg's rules, in the following these melodies will be

called nn-melodies. The rules that we applied are the following [1]:

- durational contrast
- melodic charge
- articulation of repetition
- leap tone duration
- leap articulation
- high loud
- phrase.

As an example of comparison, in figure 1 are showed the time (in milli-seconds) deviations in the two non deadpan performances of the theme of the K. 284 sonata. The 0 value is the nominal value, it corresponds to the deadpan version (i.e. no time deviations).

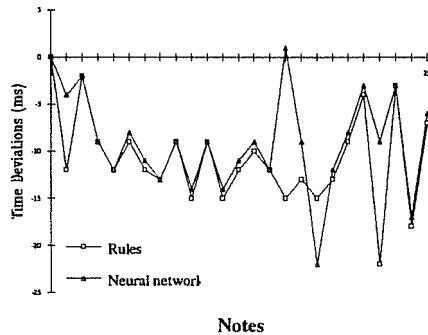


Fig. 1. Time deviations in the K 284 sonata (particular).

Equipment

The melodies were performed via MIDI by a Yamaha Disklavier Grand Piano connected to a 80386/40MHz PC compatible. To

obtain the deadpan melodies and the rules-melodies we used a program called MELODIA developed at the C.S.C. [5]. To obtain the nn-melodies we trained two neural networks: one for the loudness deviations, and another for the time deviations (see Figure 2).

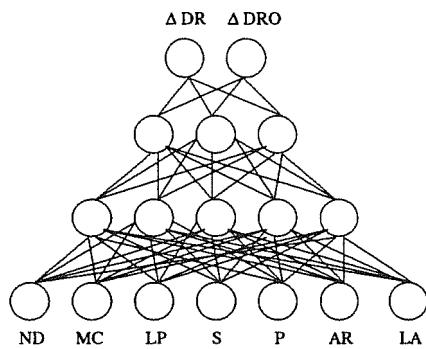


Fig. 2. Neural network for time deviations (ND= Nominal Duration; MC = Melodic Charge; LP = Leap Presence; S = Semitones in a leap; P = Phrase; AR = Articulation of Repetition; LA = Leap Articulation).

Subjects

Subjects for the study were 20 professional musicians, and students of the last years of the music conservatory of Venice, who volunteered for the experiment. They were 12 men and 8 women. The youngest was 15 years old and the oldest 32 years old. 17 of them were pianists: undergraduate and postgraduate.

Procedure

Subjects were asked before the experiment began that they had to read a paper with the instructions for the experiment and the cells where to write their judgements. Each subject was given a copy of the paper. In the paper was explained that the aim of the experiment was to compare three different piano performances built with the help of computer. In the paper were listed the title and the authors of the melodies. The text of the paper was the following:

"You will listen to three different performances of each melody, and you'll have to evaluate the musical quality of each performance as if a student is playing. You must your evaluation with a note from 1 to 10, using all the scale if possible. 1 is the worst note, 10 is the best one: avoid to give too much notes in the intermediate range (between 5 and 6), try to use extreme values (1 and 10).

The judgement doesn't have to be to critic in a absolute sense, but has to show the qualitative differences, which you find in the three different performances of the same melody. There aren't right or wrong answers: the aim of this test is to find the performance you think is the best. Between two melodies you have 30 seconds of time to judge the previous three different

performances of the melody just heard, and to write it in the opposite cells."

The three different performances of each melody were played in a random order.

The total duration of the test was 4'50".

Results

The results can be analyzed by using an ANOVA with repeated measures on each of two factors (version, melody). The analysis shows significant effects for version [$F(2,38)=10.36; p=.000$].

The nn-melody is the most preferred version (preference rating, 6.73), followed by the rules-melody (6.3) and the deadpan (4.28).

The most important result is the preference given by the subjects to the performed melodies (rules-melodies, and nn-melodies), that obtained a mean score 2 points greater than the deadpan version. Furthermore, if we consider the scolastic Italian tradition, the subjects gave a more than fair rating to the performed versions and an unsatisfactory rating to the deadpan version, even if these values don't interest the extreme values of the scale (from 1 to 10) (see Figure 3).

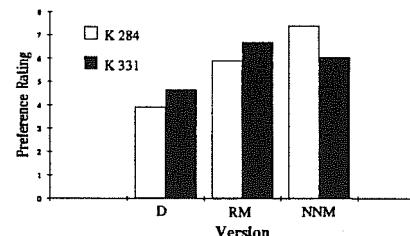


Fig. 3. Means for the interaction between the effects of version and melody (D = deadpan; RM = rules-melody; NNM = nn-melody).

Discussion

The subjects found that the greatest difficulty was the small difference between the three performances of the same melody, often they said: "They are quite equal". This fact valids the initial hypothesis, and stresses the need to continue our research considering a larger number of performing rules and a larger repertoire.

The principal outcomes of this experiment are the equivalence of the nn-melodies and the rules-melodies, and the preference they had in respect to the deadpan melodies. The reasons because the differences are not so marked are mainly two: we used $k=1$ in the Sundberg rules (this means a little emphasis in the performance rules, since all the melodies don't have a slow metronome); from the discussion we had with the subjects after the test emerged the difficulty to find great differences

between the three performances of the same melody. The mean of the subjects find the performances to be poor, maybe for the absence of an accompaniment.

Conclusions

In our opinion, the main reason for the preference of the nn-melodies in respect to the rules-melodies is that the deviations depends from the contribution of more rules. When only one of these contributions is the responsible of the deviations, then neural networks and Sundberg's rules give the same results. When more factors act together, then the additive action of Sundberg's rules system, and the properties of interpolation proper to the neural networks give different results. From this observation and from the results of the test it comes out that neural nets follow strategies which are closer to the performing action of a human performer, and so they can simulate in a better way the process of performance. (see Figure 1)

References

- [1] Friberg A. "*Generative Rules for Music Performance: A Formal Description of a Rule System*", Computer Music Journal, vol. 15, No. 2, pp. 56-71, Summer 1991
- [2] Sundberg J. et al. "*Performance Rules for Computer-Controlled Contemporary Keyboard Music*", Computer Music Journal, MIT Press, vol. 15, No. 2, pp. 49-55, Summer 1991
- [3] Bresin R., G. De Poli, A. Vidolin "*Un approccio connessionistico per il controllo dei parametri nell'esecuzione musicale*", Atti IX Colloquio di Informatica Musicale, Genova, pp. 88-102, 1991
- [4] Bresin R., De Poli G., Vidolin A. "*Symbolic and sub-symbolic rules system for real time score performance*", Proceedings of the 1992 ICMC, International Computer Music Association, San Francisco, 1992
- [5] Bresin R. "*MELODIA: a program for performance rules testing, teaching, and piano scores performing*", elsewhere in these proceedings, 1993

Timbre clustering by self-organizing neural networks

Giovanni De Poli, Paolo Prandoni, Paolo Tonella
CSC—DEI University of Padova, Via Gradenigo 6a,
35131 Padova, Italy
tel: +39-49-8287631, fax: 8287699, email: depoli@dei.unipd.it

Introduction

Timbre is that attribute of auditory sensation which allows listeners to rate as different sounds presented in ways altogether similar with respect to intensity, duration, and pitch. The similarity between two sounds can be characterized in physical and mathematical terms only with difficulty because it is a subjective attribute and it depends on a large number of parameters.

J. M. Grey, in his classic work [1], introduces the concept of "timbre space", a means with which he conveyed the vague notion of *similarity* between timbres into the precise notion of a *metric rule* in a three-dimensional space. This space was the result of a multidimensional scaling applied onto a large set of subjective similarity ratings obtained in experimental sessions. A physical interpretation of the reasons for such a spatial distribution was also provided.

In this work we will try to follow the lines of Grey's experiment, but using a neural network as the means to rate timbre differences and to transform them into metric

relations. Neural nets have been used already in this field of research [4]; the aim of our work is to simplify timbre multidimensionality, following the lines of Grey's experiment, and to obtain similar results in terms of clusterization and of timbre space. The tools we use are Kohonen self-organizing neural networks (KNN): they show an ability to correctly classify items outside the training set, and they prove highly insensitive to noise. Another reason for their use comes from neurophysiology: the principles of self-organization Kohonen proposes were derived from a model of the cerebral cortex; it is therefore interesting to compare our results with those obtained by Grey starting from subjective judgments.

Grey timbre space

J. Grey's experiments at Stanford University in 1975 were aimed at a thorough investigation in the field of musical timbre. He considered the following synthetic test sounds, obtained from a spectral analysis of recorded true instruments: bassoon (BN), normal cello (S2), E flat clar-

inet (C1), flute (FL), french horn (FH), english horn (EH), muted cello (S3), oboe (O1, O2), cello sul ponticello (S1), soprano sax (X1, X2, X3), trombone (TM), and trumpet (TP); during the experimental sessions, a group of musically trained listeners provided subjective ratings of the differences between tones. These perceptual data were averaged and arranged in a *similarity matrix*. This matrix was then processed using a multidimensional scaling (MDS) algorithm; the result was the distribution of the timbres in an n-dimensional space; at the same time, the matrix was analyzed using a hierarchical clustering algorithm based on the diameter method, and the result was an independent timbre grouping. The most interesting result was that the clusters thus obtained enclosed timbres located at low distance in the three-dimensional timbre space produced by the MDS algorithm. Grey proposed a physical interpretation for the three dimensions, showing the first dimension to be related to the spectral distribution of energy, the second dimension to the presence of synchronicity in the attack stage through the harmonics, and the third dimension to the presence of high frequency inharmonic noise with low amplitude, during the attack segment.

As timbre, in its definition, is the feature which differentiates sounds under the same conditions of pitch, intensity and duration, Grey first had to equalize the sample sounds with regard to those parameters. This equalization stage featured many psycho-acoustic sessions aimed at the comprehension of the phenomena underlying sub-

jective perception and finally produced a set of sound samples where the timbral issue was the only discriminant.

We used the same data Grey used; they consist of a line-segment approximation of the true evolutions both in frequency and in amplitude of the sound partials as they resulted from a heterodyne analysis of the equalized analog signal.

Self-organizing neural networks

Kohonen's neural networks are inspired by the process that seems to be responsible for the map-like organization of the cerebral cortex; the observable organization of the cortex neurons shows that some zones of the cortex are sensitive to certain stimulations and indifferent to others. The basic mechanism, believed to account for this process of self-organization of the brain, is called the *Hebb principle*; it asserts that if a particular neuron has a considerable reaction to a stimulation, its synapses adapt themselves to the acting stimulus, and a lateral feedback process takes place; an activity bubble is formed in the close neighbourhood of the cell, while cells surrounding the bubble are inhibited. In this way a clusterization process is generated, and the activity bubbles come to be located in different zones of the neural map according to the stimulations to which they are most sensitive.

T. Kohonen formalized this process into a simple numerical algorithm [5]. The arising neural model shows surprising properties of *self organization*: its inner structure modifies to become an n-dimensional projective model

of the m -dimensional probability space from which the input samples come. As it is generally $n < m$ (while $n = 1, 2$), the neural map performs a *feature extraction*: along the n axes of the map those input features are mapped which have the largest numerical variance. This also explains the good behavior these models exhibit in the presence of noise : KNN can maximize the amount of information stored because they organize complying to two conflicting requirements: to increase the variance of the outputs of all neurons, with the purpose of recognizing the main features of the inputs; and to introduce a certain degree of redundancy, with the purpose of obtaining correct answers even in presence of noise [6].

3D Kohonen Nets

The first experiments we carried out referred directly to Grey's main result: the three-dimensional timbre space. To obtain results to be compared directly with Grey's, we planned using a 3D neural model, extending Kohonen's equations into the third spatial dimension.

The basic algorithm ruling the self-organization process is the following: at each training step t a new input vector $x(t)$ is presented to the net; the neuron i whose inner values vector m_i is closest to the input vector x is selected as the *best matching neuron*. Different metric rules can underlie this matching criterion; for instance, we adopted the euclidean metric and the "city block distance". Around the best matching neuron a topological neighborhood $N_c(t)$ is defined as the spatial region where

the actual learning occurs: for all the indices $i \in N_c(t)$

$$m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_i(t)],$$

while the other neurons are not updated. Both N_c and α , the learning rate, decrease with time: the main structural changes in the net happen at the beginning of the process, when the neighborhood is large, while the remaining steps allow a fine tuning of the neuronal inner values. In our case the topological neighborhood is three-dimensional; its actual shape, cubic or spherical, is not essential, nor is it its shrinking rule which could be linear or exponential.

Literature, however, offers almost no example of three-dimensional forms of this equation. A mathematical analysis of KNN dynamics is extremely difficult; their properties were discovered through experimental simulation and practical applications. For this reason some preliminary experiments were performed to verify this extension.

A first task was to run the classical self-organization test [5, pag. 133] on our new structure: if the input vector x is a random variable with a stationary probability density function $p(x)$, then an ordered image of $p(x)$ will be formed onto the input weights m_i of the processing units. If an uniform distribution over a cubic region is used for the input space, an ordered cubic lattice of points should be obtained as the ultimate structure of the map. Kohonen suggested a minimal number of training steps of 500 times the number of neurons in the net [6, page 1496]; working with this too conservative an estimate. Probably, the more complex lateral interferences

in the solid case require allowing a longer phase to structural modifications.

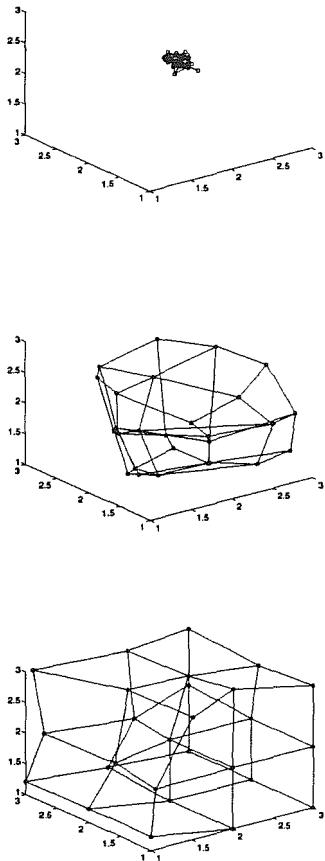


Figure 1: Three stages of evolution in the process of self ordering for a three-dimensional map. (a) startoff, (b) 1000000 iterations; (c) 2000000 iterations.

Figures 1.a, 1.b, and 1.c illustrate two significant steps of this expected evolution and show the validity of the three-dimensional model. In these plots the neuron values m_i are represented as circles

in the same coordinate system of the input values, with lines connecting those units which are adjacent in the neuronal array. Figure 1.c shows how adjacent units end up with assuming adjacent values.

3D Clusterization

At this point we presented the net with numerical data obtained directly from those used by Grey in his listening sessions. We used samples of the frequency and amplitude evolutions of the sound signal in their line-segment approximation, so that all of the processing was made by the neural network; we also tried with data outcome of a pre-processing of sounds, so that the network operated only at the most critical stage, the classification stage [8]. In all cases, the way we used Kohonen networks was somewhat fragile because only few learning samples were available with respect to the number of neurons in the network. A lack of samples causes a great sensitivity in the network final state to the starting random values of the weights. It happens that some of the neurons remain untouched by the learning process and the inner structural organization cannot unfold. It is possible to reduce this sensitivity to the initial conditions computing the mean of the different results obtained in a series of experiences, so that the effect of the initial random weights is canceled by the average [7, page 7]; we studied the convergence properties of the average, and we noted the presence of a final mean configuration with low values of variance, and, accordingly, of the relative error (3% is a typical value).

We obtained the best results using Grey's data directly: in the original line-segment representation all the necessary information to reconstruct the complete heterodine analysis of the timbre is contained and it could thus be used as an input to the neural network. We built an input file containing, for the first 20 harmonics of 12 signals, 10 amplitude envelope samples and 5 frequency envelope samples, and we fed it to a network sized $8 * 8 * 8 = 512$ neurons. After the learning phase, we computed the matrix of relative timbre distances from the spatial location of the best-matching neurons in the map; to this matrix, the same clusterization algorithm used by Grey was applied and the result was:

$$\{(BN\ FH)\ [TP\ (FL\ S2)]\ [S1\ S3]\} \\ \{[(C1\ EH\ TM)\ O2]\ X3\}$$

where the brackets split successive levels in the clusterization process. The analogies with Grey's results,

$$\{[(BN\ TP)\ FH]\ [(S2\ S3)\ (FL\ S1)]\} \\ \{C1\ (EH\ X3)\} \\ [O2\ TM],$$

are encouraging; the mismatches are due more to the different times at which grouping occurs, than to actual grouping differences.

Timbre Interpolation

The most critical point in the previous experiments was the small number of learning samples, which were just the 14 original timbres. Besides averaging results, we tried to increase the number of samples: starting from Grey's original data, and following a line exploited by Grey

himself, even though for other purposes [1, pages 75–95], [3], we considered each one of the possible couples of tones and obtained, by an algorithm of linear interpolation of the spectral envelopes, two artificial tones at $1/3$ and $2/3$ of the distance between the couple extremes. In so doing, we implicitly discarded all information about the frequency evolution of partials, adopting a coding of sounds which Grey calls the *fixed frequency model*. In the end we reached a data set of 200 units.

Clustering algorithms are generally very sensitive to little perturbations in the data points; therefore, even if the timbre space built by the net were not so much different from Grey's, the clustering algorithm could produce a completely mismatching result. Committing the accuracy judgement to a close match between clusterings seems too strict a requirement; however, since Grey does not provide the similarity matrices he used, a comparison between them and our distance matrix, which would be the best criterion, is impossible. In order to obtain a significant index of similarity for the timbre spaces, we exploited the information contained in [1, page 60], that is, the *order* in which clustering occurs: 1.(S1, S3), 2.(O1, O2), 3.(BN, TP), 4.(X1, X2), 5.(C1, C2), 6.(X3, EH), 7.(FL, S1), 8.[(BN, TP), FH], 9.[(O1, O2), TM], 10.

We define the following *index of disorder*

$$D = \sum_{k=1}^N |(d_{k+1} - d_k) - |d_{k+1} - d_k||$$

where d_k is the euclidean distance

between *net* points which, in Grey's space, belong to the k -th cluster. Distances are computed according to the diameter algorithm, that is, the maximum among all possible distances in a group. Clearly $D \geq 0$, and $D = 0$ only if d_0, \dots, d_N are in ascending order. This latter case does not imply spatial equality, but grants a good similarity. When $D > 0$, distances are in a scrambled order; each inverted group contributes to D with a term proportional to the degree of inversion.

Experiments using the extended data set were run on both two- and three-dimensional networks; convergence now required a huge number of steps and rendered our work a trial to patience. Evaluations of the index of disorder proved consistent in all of the experiences: after a first widely varying phase, due to the large structural changes occurring at the beginning of the self-organization process, the index settled around a value of fifteen (fig. 2), which indicates that few timbres are slightly misplaced. In the analysis of the index behavior, no substantial differences were found between plane and solid networks, suggesting that two dimensional structures manage to locate timbres well enough; since there is a great saving in time, the bulk of the experiments was then conducted on plane nets sized $10 * 5 = 50$ neurons.

Another topological issue came to validate our results: Grey showed how artificial timbres obtained through linear interpolation are acoustically perceived as "half way" between the two timbres from which they originate. Similarly, such an artificial timbre is mapped

by the net approximately half way along the line between the two "parents" (fig. 3); this is not obvious if we recall that KNN are *nonlinear projectors*: if linear relations in the input space are preserved, this means that the numerical form into which input samples are coded is well representative of their differences. We refer to this finding as to the inner "coherence" of the neural space.

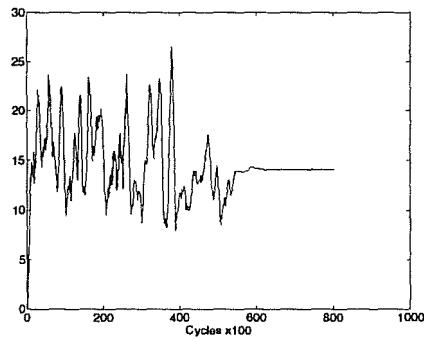


Figure 2: Typical evolution of the index of disorder

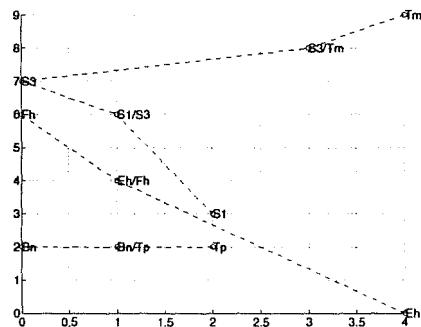


Figure 3: Neural space coherence

Conclusion

KNN are an interesting tool for the classification of a data set belonging to a space with large dimensionality, a task where classical tools for the extraction of high-variance features fail. We obtained maps which, even though different from Grey's timbre space, were not so far away. This suggests that the model underlying the artificial networks principles of self organization resembles, in a way, the features of biological neurons organization. We could ask ourselves, however, which are the legitimate expectations in such experiments. From the wide spatial separation between tones and the inner coherence of the neural space we can infer that the net is capable of efficiently handling a multidimensional feature like timbre; it would be unlikely, however, to have the same results as those obtained from a group of trained listeners. After all, Grey's model was developed in a peculiar environment, and need not be assumed as an absolute target. It would be interesting, among other things, to repeat the tests that led to it with a group of untrained people, or with sound samples of a better quality. In fact, it should be noticed that Grey's synthetic tones are of a low sound quality; future developments will surely profit of a higher quality sampling of the test timbres, and of an adequate signal pre-processing.

With regard to the data reduction techniques, deeper studies are under completion at Padova University; the best results have been obtained using pre-processing based upon Grey's observations, while Charbonneau's methods gave worse final configurations. A development

of this work is the substitution, at the initial stage in the process of timbre recognition, of the heterodyne analysis with a simulator of the human ear; in this way the operations made on input signals by biological organs and neurons is entirely reproduced by an artificial system.

References

- [1] Grey J.M., *An exploration of musical timbre*, Rep. STAN-M-2, Stanford University, 1975.
- [2] Grey J.M., *Multidimensional perceptual scaling of musical timbres*, J. Acoust. Soc. Am., 61(5): 1270-1277, 1977.
- [3] GordonJ.W., Grey J., *Perception of spectral Modifications on Orchestral Instrument Tones*, Comp. Music J., 2(1): 24-31, 1978.
- [4] Feiten B., Frank R., Ungvary T., *Organizations of sounds with neural nets*, Proc. ICMC 91, p. 441-444, 1991.
- [5] Kohonen T., *Self-organization and associative memory*, Springer V., Berlin, 1984.
- [6] Kohonen T., *The Self-Organizing Map*, Proc. of the IEEE, 78(9): 1464-1480, 1990.
- [7] *The Self-Organizing Map Program Package*, Helsinki University of Technology, 1992.
- [8] De Poli G., Tonella P., *Self-Organizing Neural Networks and Grey's Timbre Space*, Proc. ICMC 93, 1993.

NEURAL NETWORKS AND STYLE ANALYSIS: A Neural Network that Recognizes Bach Chorale Style

Margaret Johnson

Department of Computer Science
Stanford University
Stanford, CA 94305-3068
johnson@cs.stanford.edu

I. Neural Networks in Music

A *neural network* is an analysis tool that is very loosely modeled on the structure of the human brain. It is composed of elements that imitate the most elementary functions of a biological neuron. These elements are organized in a way that generally resembles the anatomy of the brain. Despite this superficial resemblance, neural nets exhibit a surprising number of the brain's characteristics, although much smaller in scope. For example, they seem capable of learning specific patterns after training. They also appear to generalize from previous examples to new ones, and they can often abstract the essentials from inputs containing irrelevant data.

In recent years, neural networks have been applied to music in various ways. For example, networks have been trained to "compose" music in a particular

style (Kohonen's work [1], Mozer's CONCERT system [2], or Todd's algorithmic composition network [3]). Networks have also been used to classify rhythms (David & Sandon's Tempnet [4]); for pitch perception (Sano & Jenkins [5], or Bharucha & Todd [6]); for tonal analysis (Scarborough, Miller & Jones [7], Laden & Keefe [8], or Bharucha [9]); to determine optimal fingerings for string instruments (Duff [10], or Sayegh [11]); for understanding musical perception (Bharucha [12]); even for the study of jazz and improvisation (Baggi's Neurswing [13] and Alpaydin [14]).

II. Project Description

The present project represents an exploration of the uses of neural networks in style analysis. Style analysis is defined as the identification of characteristic features in the music of a composer by comprehensive

analysis of harmony, rhythm, melody, sound and form. We assess the potential for using a neural network as a general tool in identifying key features of a composer's style.

We construct a neural network, and train it on patterns derived from the analysis of a set of Bach chorales, and a set of chorales taken from the 1940 Hymnal [15]. Another term for a chorale is a hymn, which is a religious song of praise characterized by a chordal style in four parts, suitable for congregational singing. The 1940 Hymnal is a primary collection of hymns used in Protestant Episcopal churches. Included are hymns from medieval times to the present, including many works by Bach.

Once the neural net is trained on the two sets of patterns, we test it to see if it can recognize whether or not a chorale is written by Bach, when presented with any chorale (in a major key) from the 1940 Hymnal. In addition, we analyze the resulting values in the network to see what happened during the training. If we can understand what the neural net learned during training, we may be able to learn something about Bach's compositional style in composing chorales.

Details of Implementation

The neural network for this project is a standard back-

propagation model with 23 inputs, 3 neurons in the hidden layer, and 1 output. The output is a real number in the 0.0 to 1.0 range. If the output is close to 1.0, the input pattern represents a chorale similar in style to Bach. Outputs closer to 0.0 represent chorales of other composers and styles.

The inputs are a series of descriptive categories. This information is encoded and organized, or "pre-processed", to make training easier. The encoding is sometimes binary (1 if true and 0 if false), and other times a real number. Note that binary values are 0.99 and 0.01 rather than 1 or 0 because the use of 0 in the neural network calculations complicates the training process. The categories are given below:

- 1) Meter: 0.99 if 4/4, 0.5 if 3/4 and 0.01 if anything else.
- 2) Key: The key is encoded as a real number based on the following scale. These are the only keys found in the analyzed Bach chorales:
G A D F Bb C Eb E Ab
.9 .8 .7 .6 .5 .4 .3 .2 .1
The more common keys are assigned higher values. If a chorale from the 1940 Hymnal is in a key other than those given above, it is encoded with 0.001.
- 3) Chord Frequency: Based on a complete harmonic analysis, we determine the number of times each chord is used. The chords

we count are: I, ii, iii, IV, V, vi, any vii^o, any secondary dominant, and inversion chords.

4) Non-harmonic tones: We count how many non-harmonic tones are used in the entire chorale. These are tones that are not a part of the chord currently playing. The actual value used in the input pattern is the frequency. For example, if there are 112 notes in the entire piece and 16 of these are non-harmonic tones, the value of this input is 0.143.

5) Unaccented passing tones: In addition to counting the total number of non-harmonic tones, we also count the number of unaccented passing tones.

6) Cadences: We count the total number of cadences, and the frequency of cadences on I, ii, IV, V, and vi. So, if there are 8 cadences in the piece and 3 are cadences on I, the value for the I category is 0.375.

7) Modulations: We look for modulations to the keys of ii, IV, V, and vi. If we find them, we assign a weight based on the duration of the modulation. If a modulation extends over more than two measures, we are firmly based in the new key, so this modulation would be assigned 0.9. If a modulation extends over 1/2 measure, this is a brief excursion (if really a modulation at all), so we assign 0.1. In between, we assign specific values for different

durations.

8) Length: The longest chorale we analyzed was 88 measures, so we compare each chorale to this maximum. If a chorale is 44 measures long, it has a length value of 0.5.

The set of input patterns consists of the first 85 Bach chorales from the 371 Four-Part Chorales [16], and 85 chorales from the 1940 Hymnal, starting with number 275. We analyze only those in major keys. This set of 170 patterns is used to train the network. We use the standard back-propagation training algorithm [17].

The set of input patterns are presented to the network 4000 times, at which point the error level is 0.008. This indicates that the network has learned all of the patterns with only a slight chance of error. We go through several iterations of this process changing various parameters of the network. The primary parameter that we study is the number of hidden neurons. In neural networks, it is essential to determine the appropriate number of hidden neurons. If we have too few neurons, the network will not train at all. In the present project, we were unable to train a network with just one hidden neuron. If we have barely enough neurons, the network may train, but it might not be robust in the face of irrelevant data, or it won't

recognize patterns that it has not seen before. In this project, this was the case with a network of two hidden neurons.

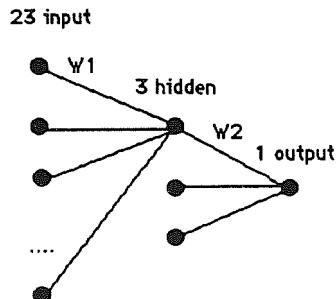
Testing

In order to determine that the network really has learned something more than just the specific patterns in the training set, we run additional tests using chorales not included in the training set. We encode 10 chorales from Bach and the 30 from the 1940 Hymnal (including 5 by Bach). The error level for this test set is 0.005, indicating that the patterns are recognizable. The output of the network for these 40 patterns is correct in all cases (close to 1.0 for Bach and close to 0.0 for chorales not composed by Bach).

III. Analysis of Trained Network

The weights in a neural network are modified during training until we end up with values that allow the network to recognize patterns and classify them correctly. These values may represent what the network actually learned during the training. Thus, an analysis of the weight values in the trained network may provide clues to the "essence" of Bach's chorale style.

A diagram of the structure of the network appears below:



The values in this network consist of the following:

- 1) The 170 input patterns (85 Bach and 85 from the 1940 Hymnal) each with 23 inputs.
- 2) The weights in the trained network connecting each input to each of the three hidden neurons. In the diagram this is weight vector W_1 , of which we show only the weights from each input going into the first hidden neuron ($W_{1,1}$ to $W_{1,23,1}$). The network is fully connected, so there are 23 connections going into each hidden neuron (i.e., $W_{1,2}$ to $W_{1,23,2}$, $W_{1,3}$ to $W_{1,23,3}$). We do not show all these other connections in the diagram.
- 3) The values in the hidden neurons. The hidden neurons each have a specific value when a particular input pattern is presented to the network. This value is the sum of the inputs times the corresponding weights coming into the neuron. A sigmoid function (this function is commonly used in the back-propagation training algorithm) is applied to this sum to provide the actual value of the hidden

neuron. This value is between 0 and 1.

4) The weights connecting each hidden neuron to the output. This is W2 in the diagram; it has three components (W2 1,1, W2 2,1, W2 3,1).

To do the analysis, we do not look at the specific weights in W1 and W2, but rather at what is termed the *weight-state vectors* [18]. This is the value of the product of the input times its corresponding weight when an input pattern is presented to the network. Just looking at the weights themselves does not give as much information as looking at different patterns of the weight-state vectors when input patterns are applied.

Each input pattern has 3 weight-state vectors with 23 components, one set for each hidden neuron. For example, the weight-state vector for the first hidden neuron (shown in the diagram) would consist of the following 23 components: Input 1 * W1 1,1; Input 2 * W1 2,1; Input 3 * W1 3,1 etc.).

The basic idea of the analysis is to start with a set of chorales which are most typical of Bach (have an output of near 1.0). Then, we work backwards through the network analyzing the values of the weight state vectors of W2, the values of the hidden neurons, and finally, the weight state vectors of W1,

where we hopefully learn which specific inputs are causing an output near 1. These inputs represent the distinguishing characteristics of Bach chorales as learned by the network.

The analysis proceeds as follows:

- 1) Determine which chorales have the highest output, i.e., the output closest to 1.
- 2) Look at the W2 weight-state vectors, to determine which hidden unit(s) are critical to obtaining the output.
- 3) Look at the specific hidden unit value(s), as determined in step 2, for each of the chorales obtained in step 1.
- 4) Look at the weight-state vectors for the specified hidden unit(s)

and the specified chorales to determine which inputs excite or inhibit the hidden unit(s).

The following chorales from the 371 Four-Part Chorales, have outputs closest to 1.0: 5, 20, 24, 64, 84, 102, 124, 135, and 179. An analysis of the weight-state vector W2 indicates that the first hidden unit is critical, specifically, this hidden unit must be a large value (greater than 0.9) in order for the output to be near 1. This being the case, we look at the W1 weight-state vectors for large positive values. The inputs associated with these values are the ones that excite the hidden unit to larger values.

We find that the inputs having the strongest effect on the first hidden unit are: Meter, Key, Non-Harmonic Tones, Unaccented Passing Tones, Cadences on I, Frequency of Secondary Dominants. Those inputs having a lesser effect, but still a positive one are: Cadences on ii, Cadences on vi, Modulations to ii, Modulations to vi, Frequency of ii chords, Frequency of vi chords. These inputs represent the characteristics that distinguish a Bach chorale from one by another composer. Further analysis of the specific values of the inputs for the set of chorales with the highest output provides more information:

- 89% are in 4/4
- 89% are in the keys of D,A or G
- 79% have non-harmonic tone frequency of 12%
- 79% have unaccented passing tone frequency of 9%
- 89% cadence half the time on I
- 89% have secondary dominant frequency of 12%
- 79% cadence 6% of the time on ii
- 79% cadence 12% of the time on vi
- 79% have ii modulation strength of 0.23
- 79% have vi modulation strength of 0.33
- 79% have ii chord frequency of 8%
- 79% have vi chord frequency of 10%

What happens during training is each time a chorale with these characteristics appears as an input pattern (which is quite often), the weights connecting the inputs specified above to the first hidden unit are strengthened. Later during testing, when a chorale with these characteristics appears, the first hidden unit is excited to the point that it produces an output near 1. When a chorale without these characteristics is presented, the effect of the first hidden unit is negated by the effects of the second and third hidden units, causing the output to be near 0.

We perform one more test to determine if the characteristics listed above are typical of Bach chorales. To do this, we analyze the entire set of 85 Bach chorales. This is done with a computer program that processes the input pattern file. First, the average score for each of the inputs is determined. Then, we find what percent of the chorales is in close proximity to the average. The results are close to those given above:

- 88% are in 4/4
- 70% are in the keys of D, A or G
- 72% have non-harmonic tone frequency of 11%
- 67% have unaccented passing tone frequency of 8%
- 83% cadence half the time on I

- 85% have secondary dominant frequency of 11%
- 83% cadence 5% of the time on ii
- 83% cadence 10% of the time on vi
- 71% have ii modulation strength of 0.20
- 84% have vi modulation strength of 0.29
- 68% have ii chord frequency of 7%
- 71% have vi chord frequency of 10%

Based on these results, we conclude that the network did indeed learn the characteristics which distinguish Bach chorales from those written by other composers.

IV. Conclusion

It does not come as a surprise that the distinguishing characteristics of Bach chorales are those discovered in this project. Anyone who hears both a Bach chorale, and perhaps one by Thomas Tallis, knows there are more non-harmonic tones, more chord variety, and more interesting modulations.

The value of this project is in determining that a neural network is capable of recognizing distinctive features of a composer's style. The next step is to make the process more dependent on the network and less on the "pre-processor" who analyzed the chorales and created

the patterns. Now that we know neural networks can provide insight into a composer's characteristic style, we can experiment with less precise inputs to further test a neural net's usefulness as a style analytic tool.

References

1. T. Kohonen, "A Self-Learning Musical Grammar or Associative Memory of the Second Kind," *Proc. Int'l Joint Conf. Neural Networks*, Vol. 1, 1989, p.1.
2. M. Mozer, "Simulating Melodies in Style of Bach," *Computing in Musicology*, 1991, p. 54.
3. P. Todd, "A Connectionist Approach to Algorithmic Composition," *Computer Music Jour.*, Vol. 13, No. 4, 1989, p. 27.
4. I.L. David and P. Sandon, "Temporally Sensitive Neural Networks," *Proc. Int'l Joint Conf. on Neural Networks*, Vol. 2, 1991, p. 2104.
5. H. Sano and B.K. Jenkins, "A Neural Net Model for Pitch Perception," *Computer Music Jour.*, Vol. 13, No. 3, 1989, p. 41.
6. J. Bharucha and P. Todd, "Modeling the Perception of Tonal Structure with Neural Networks," *Computer Music*

- Journal*, Vol. 13, No. 4, 1989, p. 44.
7. D. Scarborough, B. Miller, and J. Jones, "Connectionist Models for Tonal Analysis," *Computer Music Journal*, Vol. 13, No. 3, 1989, p. 49.
8. B. Laden and D. Keefe, "The Representation of Pitch in a Neural Network Model of Chord Classification," *Computer Music Jour.*, Vol. 13, No. 4, 1989, p. 12.
9. J. Bharucha, "MUSACT: A Connectionist Model of Musical Harmony," *Proc. Cognitive Science Society*, Hillsdale, NJ: Earlbaum Press, 1987.
10. M.O. Duff, "Backpropagation and Bach's Fifth Cello Suite (Sarabande)," *Proc. Int'l Joint Conf. on Neural Networks*, Vol. 1, 1989, p. 121.
11. S. Sayegh, S. "Fingering for String Instruments with Optimal Path Paradigm," *Computer Music Jour.*, Vol. 13, No. 3, 1989, p. 76.
12. J. Bharucha, "Music Cognition and Perceptual Facilitation: A Connectionist Framework," *Music Perc.*, Vol. 5, No. 1, 1987, p. 1.
13. D. Baggio, "Neurswing: An Intelligent Workbench for the Investigation of Swing in Jazz," *Readings in Computer Generated Music*, edited by Denis Baggio, Los Alamitos, CA: IEEE Computer Society Press, 1992.
14. R. Alpaydin, "Connectionist Approach to Improvisation Using Fingering Pattern Association," M.S. Thesis, Bogazici University, Istanbul, Turkey, 1992.
15. *The Hymnal of the Protestant Episcopal Church*, New York: Church Pension Fund, 1940.
16. J.S. Bach, *Four-Part Chorales*, Wiesbaden: Breitkopf & Hartel.
17. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing, Exploration in the Microstructure of Cognition*, 3 volumes, Cambridge: MIT Press, 1986.
18. R. Gorman and T.J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Networks*, Vol. 1, 1988, p. 75.

MODELLING HARMONY-BASED JAZZ IMPROVISATION: AN ARTIFICIAL NEURAL NETWORK APPROACH

Petri Toivainen

University of Jyväskylä
Department of Musicology
PL 35
SF-40351 Jyväskylä (Finland)
fax +358 41 601 331
E-mail ptoiviai@tukki.jyu.fi

Introduction.

In cognitive science and research of artificial intelligence, there are two central paradigms: symbolic and analogical. In the symbolic approach, the cognitive models are based on entities, which are symbols both semantically (referring to external objects), and syntactically (being part of the rule-based manipulation of symbols). In the analogical approach, one tries to imitate the cognitive phenomena using some other system. Within the analogical paradigm, artificial neural networks (ANNs) have recently been successfully used for the modelling and simulating of cognitive phenomena. ANNs are non-linear dynamical systems consisting of a large number of massively interconnected simple processing elements, or neurons. ANNs are parallel (the neurons interact essentially simultaneously and independently) and distributed (their knowledge is represented

by the connection strengths between the neurons; the cognitive entities are represented by activation patterns of the neurons). The most prominent feature of ANNs is their ability to learn by example, and, to a certain extent, generalise what they have learned.

Improvisation, the art of spontaneously creating music while playing, is the basic element of nearly all musical cultures of the world. In the western tradition, the art of improvisation reaches its highest level among jazz musicians. In jazz, one improvises on melodic, harmonic, as well as rhythmic level. The basis of improvisation differs considerably between different styles of jazz: a *Dixieland* musician bases his improvisation, more than the others, on the structure of the melody of the composition; a *bebop* musician leans rather on the harmonic structure; whereas a *free jazz* group, for example, may operate solely within the framework of some prearranged

musical structure.

As a cognitive process, the improvisation of jazz is extremely complicated. Irrespective of the jazz style in question, the improviser has to take into account constraints on various hierarchical levels. Every musician has, too, a personal way of approaching this problem.

The art of improvisation is learned mostly by example. Instead of memorising explicit rules, the student mimics the playing of other musicians. On the basis of this material, he then forms implicit rules for the style of improvisation concerned. This kind of learning procedure cannot be easily modelled with rule-based expert systems. On the other hand, artificial neural networks, or connectionist systems, are suitable for this purpose. Neural networks are systems consisting of massively interconnected simple processing elements, or neurons. Their most prominent feature is the ability to learn by example.

Recently, neural networks have been successfully used for the modelling and simulating of cognitive phenomena of music (cf. e.g. [1][2][3]). In this paper, a connectionist model of bebop-style melodic improvisation, based on harmony, is described; some results, achieved by simulating the model, are presented. The basis of the model is the *target note technique*, one of the cornerstones of bebop-style jazz improvisation.

About harmony-based improvisation.

There are several factors influencing the structure of an improvised jazz melody: e.g., the harmonic structure, the melody of the theme, and the style of the accompaniment. Among the essential factors are, also, the instrument used – its fingering and range of pitches ; the musical background of the soloist; and the musical interaction with the other players of the group during the playing.

The target note technique [4][5], a common way of explaining the microstructure of an improvised jazz melody, can be described as follows: the notes of a tetrachord and its upper structure (9, 11, 13) are regarded as *principal tones*; when approaching a chord, one of its principal tones is chosen as a target note; the target note is reached through a learned melodic pattern.

For the simulation model, the target note technique can be described as a dynamic process with feedback, as presented in fig. 1. At the beginning of the process, some starting notes – here two – are fixed, as well as the present and following chords. The starting notes, together with the present chord, determine the possible melodic patterns, whereas the following chord determines the possible target notes. From the interaction of those constraints a new melodic pattern emerges. The target notes of the pattern are then

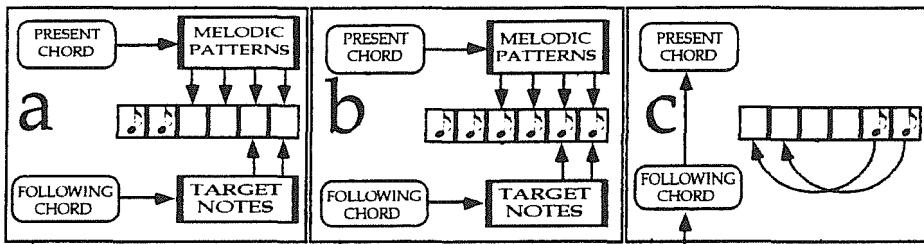


Figure 1. The target note technique as a dynamic process with feedback. Cf. text.

used as the starting notes of the next pattern.

The architecture of the model.

An essential step in constructing a model for the simulation of the production of music, is to decide how time is represented. When using neural networks, a natural choice is to use a representation analogical to “piano roll” notation, used in the old player pianos. In “piano roll” notation, time has been translated into an ordered spatial dimension. Accordingly, in neural network, the melody can be represented in a two-dimensional grid of neurons, where one dimension corresponds to time, and the other to pitch. In the improvisation model in question, the network processes one melodic pattern at a time, using a modified “piano roll” representation.

The rhythmic structure of both the melody and harmony of bebop jazz is very regular – if the phrasing is ignored. Firstly, an overwhelming majority of all jazz composed and performed is in 4/4

time [6]. Secondly, it can be said that all jazz from 1900 to the present day has employed the following ratio of time values: a quarter-note pulse, representing the *rhythmic* unit of a jazz composition; a half-note pulse, representing the *harmonic* unit; and an eighth-note pulse, representing the *melodic* unit [7]. Consequently, it is justified to make the following simplifications concerning the architecture of the network: (1) only 4/4 time is possible; (2) the network processes one half-measure-long melodic pattern at a time – faster chord progressions are not possible; and (3) the smallest possible time value is an eighth-note.

For the representation of pitch in neural networks, there are several possibilities [8]. In this model, *invariant pitch class representation* is used. The pitches are, thus, represented relative to the root of the present chord.

In the network, the melody is represented as follows: for every eighth-note, 12 neurons represent the pitch classes of the chromatic scale, relative to the root of the present chord. Moreover, one

neuron represents the rest, and another the ligature. In order to simplify the model, the following restrictions are made: (1) the contour of the melody is ignored; (2) The network produces only monophonous melodies; (3) Phrasing, dynamics, and octave are ignored. In the network, one eighth note corresponds, thus, to a group of 14 neurons. Because of syncopation, characteristic of jazz music, the target note is often played on the eighth note preceding the first beat. Therefore it is reasonable to use six groups of neurons: the first corresponding to the eighth note preceding the half of the measure in question; the following four corresponding to the half of the measure in question; and the last corresponding to the eighth note following the half of the measure in question. The network consists, thus, of $6 \times 14 = 84$ neurons altogether. The neurons of the network are fully interconnected. There are two types of connections: (1) neurons belonging to the same group (column) have fixed inhibitory interconnections; this guarantees that, in the relaxed state, only one neuron is active in every group. (2) neurons belonging to different groups have excitatory interconnections, which are modified during the learning phase, according to the *Hebbian learning rule* [9]. The network is, actually, a modified *auto-associator*.

The consideration of the type

of the present chord is carried out by feeding external activation into the auto-associator. For a given chord-type, the most often used notes receive the strongest external activation. The consideration of the possible target notes, within the melodic pattern being produced, is carried out by feeding external activation into the last two groups of neurons of the auto-associator: the neurons representing the most often used target notes receive the strongest external activation.

During the learning phase, one melodic pattern at a time is presented to the network: the neurons representing the notes of the melodic pattern are activated, together with the neurons representing the roots and the types of the present and the following chord. The learning occurs in two distinct groups of neurons: (1) the melodic patterns are learned through strengthening the connections between the active neurons of the auto-associator, and the neuron representing the type of present chord, according to the Hebbian learning rule; (2) the target notes are learned through strengthening the connections between the neurons representing the type of the following chord, and the neurons representing the invariant target notes. Inside these two groups of neurons, differing learning rates can be used, resulting in melodies of different styles. In fig. 2, the architecture of the whole network, with an example of the representation and

learning of one melodic pattern, is presented.

During the testing phase, the activations of the neurons representing the root and the type of the present and the following chord are set to maximum. The starting note of the melody is chosen, and the corresponding neuron is activated. During the relaxation process, the activations of the neurons are updated *asynchronously*: the activation of one – randomly chosen – neuron is updated at a time. The updating procedure consists of: (1) calculating the income activation, according to the formula

$$\text{input}_j = a_j + \sum_{i \neq j} w_{ij} a_i ,$$

where w_{ij} is the connection strength between neurons i and j , and a_i is the activation value of neuron i ; and (2) calculating the new activation according to the formula

$$a(\text{input}) = \begin{cases} 0, & \text{if input} \leq 0, \\ \text{input}, & \text{if } 0 < \text{input} \leq 1, \\ 1, & \text{if input} > 1 \end{cases}$$

The updating procedure continues until the network has reached a stable state. In every group of neurons, there is now one active neuron. These form the melodic pattern the network has produced. After feeding back the last two notes of the melodic pattern as the starting notes of the next pattern, and updating the chords, the relaxation process is again, started.

Observations about the experiments.

The network was taught solos played by trumpet player Clifford Brown. The material consisted of excerpts, typically 32 measures long, from six solos, taken from [10]. After the learning phase, the network was tested with *Rhythm Changes* chord progression, one of the most widely used harmonic structures in bebop. When teaching the network, various learning rates were used for the melodic patterns and the target notes. It was found that the ratio of the learning rates had a considerable influence on the style of the melodies produced. When using proper values for the learning rates and inhibition, the network was found to be capable of producing stylistically fairly consistent melodies – on the micro level. On the other hand, it cannot deal with larger structures, like melodic phrases. In this respect, the melodies produced by the network resemble those of a beginning improviser. An interesting aspect is, that the model has a sort of creativity: it can produce new melodic patterns, based on the patterns it has learned. In fig. 3, examples of the melodies, produced by the network, are presented.

Conclusions.

Above, an artificial neural network model describing the learning of melodic jazz

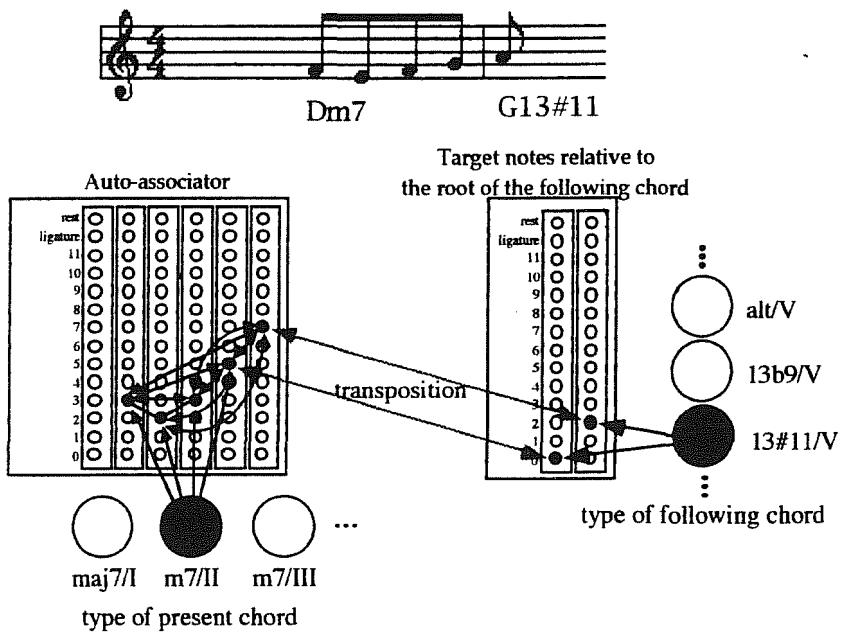


Figure 2. An example of the representation and learning of one melodic pattern in the network. The black arrows represent the connections, the strengths of which are increased.

improvisation is presented. The starting-point of the model is learning by means of examples: the student is presented melodic patterns, and after having learned them, he is able to use them in a given context, i.e., harmonic progression, and to generalise what he has learned. Instead of presenting the student explicit rules concerning the style of improvisation in question, he himself builds implicit rules, based on the material learned, and applies these in new situations.

The simulations made showed that the model is able to apply the material it has learned to a

new context. If it cannot find a proper melodic pattern for a given chord progression, it can create a new pattern, based on the patterns it has learned. This kind of adaptability – or creativity – is a direct consequence of the connectionist paradigm. The structure of each melodic pattern is not presented symbolically, but rather resides in a distributed form in the network – in the connection strengths between the neurons. To construct a rule-based expert system behaving in the same manner would be laborious. The epistemological relevance of such a system, based on numerous

explicit rules, would also be small.

The main shortcoming of the system described above is the inability to operate on higher hierarchical melodic levels, e.g. phrases. This might be overcome by, for example, using a higher-level network. This network would learn and produce only the target notes of a longer harmonic progression; e.g., four measures long. Feeding the target notes produced by this network into the lower-level network, and taking into account the respective position in the four-measure structure, would probably yield better results.

References.

[1] J. Bharucha, P. Todd: "Modeling the Perception of Tonal Structure with Neural Nets", Computer Music Journal, Vol. 13, N. 4, pp. 44-53, 1989.

[2] P. Todd: "A Connectionist Approach to Algorithmic Composition", Computer Music Journal Vol. 13, N. 4, pp. 27-43, 1989.

[3] M. Leman: "Tonal context by pattern integration over time", Informatica Musicale, pp. 21-39 Genova: AIMI, DIST, 1991.

[4] J. Mehegan: "Jazz Improvisation I: Tonal and Rhythmic Principles", pp. 127-131, Watson-Guptill Publ., 1959.

[5] S. Berg: "Jazz Improvisation: the Goal Note Method", Lou Fischer Publ., 1990.

[6] J. Mehegan: "Jazz Improvisation II: Jazz Rhythm and the

Improvised Line", pp. 17-22, Watson-Guptill Publ., 1962.

[7] J. Mehegan: "Jazz Improvisation II: Jazz Rhythm and the *Improvised Line*", p. 22, Watson-Guptill Publ., 1962.

[8] J. Bharucha: "Pitch, Harmony, and Neural Nets: A Psychological Perspective", in "Connectionism and Music", P. Todd & G. Loy ed., MIT Press, 1991.

[9] J. McClelland, D. Rumelhart: "Explorations in Parallel Distributed Processing", pp. 84-86, MIT Press, 1988.

[10] D. Baker: "The Jazz Style of Clifford Brown. A Musical and Historical Perspective", Studio P/R, 1982.

a)

Bmaj7/I Gm7/VI Cm7/II F13⁹ Dm7/III Galt Cm7/II F13⁹¹¹

Fm7/II B¹³⁹¹¹ Emaj7/IV A¹³⁹¹¹ Dm7/III Galt Cm7/II F13⁹¹¹

Bmaj7/I Gm7/VI Cm7/II F13⁹ Dm7/III Galt Cm7/II F13⁹¹¹

Fm7/II B¹³⁹¹¹ Emaj7/IV E⁹ Dm7/I Bmaj7/I

b)

Bmaj7/I Gm7/VI Cm7/II F13⁹ Dm7/III Galt Cm7/II F13⁹¹¹

Fm7/II B¹³⁹¹¹ Emaj7/IV A¹³⁹¹¹ Dm7/III Galt Cm7/II F13⁹¹¹

Bmaj7/I Gm7/VI Cm7/II F13⁹ Dm7/III Galt Cm7/II F13⁹¹¹

Fm7/II B¹³⁹¹¹ Emaj7/IV E⁹ Dm7/III Galt Cm7/II F13⁹¹¹

Figure 3. Examples of melodies produced by the network on the *Rhythm Changes* chord progression. The network was taught excerpts from Clifford Brown's solos on a) "All the things you are" and "Gertrude's bounce", b) "Confirmation" and "Donna Lee". The rectangles indicate melodic patterns which occur in the learning material. Letters "A", "G", "C" and "D" refer to the names of the compositions.

Capitolo 5

ELABORAZIONE NUMERICA DI SEGNALI

UNA NUOVA TECNICA DI SINTESI ADDITIVA BASATA SULLA TRASFORMATA INVERSA DI FOURIER

M. Barutti, G. Bertini

Istituto di Elaborazione della Informazione-CNR
Via S.Maria,46, I-50126 Pisa (Italy)
Fax +39 050 554342 E-mail:bertini@iei.pi.cnr.it

Abstract

Additive synthesis can be considered the most powerful and general synthesis technique, but it has very high computational costs. Sound synthesis systems which use massive and exclusive additive synthesis are now available on the market; however, their cost is much higher than that of systems providing an equivalent degree of polyphony based on other synthesis techniques.

The method here presented allows to generate complex musical signals using the Inverse Fourier Transform with a reduced number of harmonics, thanks to a mechanism by which further sine waves can be added. Simulation and test of complex signals synthesis have been carried out with good results. An evaluation of a system having several thousand sine waves with a good frequency and time resolution and a SNR higher than 80dB, leads to a computational cost of less than one tenth of systems using digital oscillators.

Introduzione

Nell'ambito delle tecniche per la generazione di segnali musicali è noto che la sintesi additiva è quella che permette una grande flessibilità nella fase progettuale del suono (scelta del timbro, ecc.) e consente una certa intuivita' dell'effetto della variazione dei parametri del segnale ed il suono ottenuto. Il grosso svantaggio è la notevole mole di calcolo necessaria per la sua implementazione. Grazie al continuo sviluppo della tecnologia elettronica cominciano ad apparire sul mercato alcuni sistemi che impiegano massicciamente, la sintesi additiva [1, 2]. Il costo degli apparati rimane comunque alto, più di un ordine di grandezza superiore ai sistemi di pari grado di polifonia che usano altre tecniche di sintesi.

Utilizzando le operazioni di analisi e sintesi di segnali basate sulle trasformata di Fourier su brevi intervalli di tempo, è possibile generare digitalmente dei tratti di segnale utilizzando la FFT Inversa (IFFT), valutata a partire da ampiezza e fase di

determinate armoniche. Anche questo procedimento ha un alto costo computazionale, data l'elevata risoluzione di frequenza necessaria per una buona qualità nella sintesi di segnali musicali. Di recente sono stati proposti dei metodi basati su FFT inversa nei quali sono introdotte delle semplificazioni che ne rendono possibile l'implementazione [3]. Una soluzione proposta da M. Barutti [4] (domanda di brevetto n. PI93A000003), permette di generare dei segnali complessi partendo da IFFT con un numero ridotto di armoniche grazie alla possibilità di aggiungere altre componenti con frequenza qualsiasi, tramite un meccanismo detto "metodo degli sfasamenti". Nel seguito si illustra il principio su cui si basa il metodo, tenendo conto che nell'implementazione effettiva sono usate delle ottimizzazioni che lo rendono conveniente dal punto di vista computazionale.

Requisiti della sintesi additiva.

Un dispositivo che implementi la sintesi additiva oltre a soddisfare tutti i requisiti imposti anche dalle altre tecniche di sintesi [5], (tasso di campionamento, velocità di aggiornamento dei parametri, rapporto segnale/rumore, ecc.) deve poter generare un elevato numero di onde sinusoidali contemporaneamente.

Dai risultati di studi effettuati sul suono di strumenti ad arco ed a fiato (vedi ad es. in [6]), si può supporre che 30 sinusoidi siano

sufficienti per generare singoli suoni di ottima qualità e varietà timbrica: considerando il grado di polifonia offerto attualmente dagli strumenti elettronici operanti in tempo reale con un buon livello di prestazioni, sono necessarie alcune migliaia di onde sinusoidali.

Per quanto riguarda la velocità di aggiornamento dell'ampiezza e della frequenza, partendo da precedenti realizzazioni [7] [8], sono state da noi effettuate ulteriori verifiche utilizzando schede con DSP fixed e floating point e particolari procedure di test [9]; è risultato che un ottimo valore per aggiornare la frequenza può essere compreso tra 2-3 msec, mentre per l'ampiezza è preferibile un aggiornamento più rapido, al limite uguale al tasso di campionamento.

Metodi basati sulla trasformata di Fourier

La sintesi additiva viene generalmente realizzata con il metodo dell'oscillatore digitale a lettura tabellare; l'elevato costo computazionale è dovuto al fatto che i valori istantanei di ogni sinusoide, vengono calcolati per ogni singolo campione generato, ad es. 44.1 mila volte al secondo. Un altro possibile metodo per ricavare un segnale come somma di un elevato numero di onde sinusoidali è quello di utilizzare l'algoritmo della IFFT (Inverse Fast Fourier Transform) [10]. Il segnale finale viene ottenuto giustapponendo segmenti(frames) ricavati dalla sequenza di una (o

più) IFFT e calcolati ad intervalli di tempo non superiori a 3 msec. Si ricorda che se F_0 è la risoluzione in frequenza della FFT, la frequenza della riga j -ma e il numero delle righe (o armoniche) su cui viene calcolata la IFFT sono legate dalle relazioni seguenti:

$$F_{armj} = jF_0 \quad \text{con}$$

$$j=0 \dots N_{arm}$$

$$F_0 = F_c / N_{ifft},$$

$$N_{arm} = N_{ifft} / 2$$

dove N_{arm} è il numero delle righe della IFFT, N_{ifft} è il numero di punti della IFFT e F_c è la frequenza di campionamento.

L'inconveniente di generare dei segnali tramite l'uso di una IFFT è che non si riescono a generare segnali contenenti onde sinusoidali con frequenza diversa da F_{0j} . Un'altro inconveniente di questa tecnica è che la giustapposizione dei frames può provocare brusche discontinuità in ampiezza; perché ciò non accada vengono implementati opportuni meccanismi di interpolazione (mix lineari, senquadro ecc.), noti in letteratura [11].

Il costo computazionale dei metodi utilizzanti l'algoritmo della IFFT è costituito da quello per il calcolo delle IFFT e per il mix finale dei segnali risultanti, più una quantità variabile legata alla preparazione dei parametri di partenza, cioè ampiezza e fase delle armoniche in funzione del segnale da generare.

Metodo degli sfasamenti.

Il meccanismo che nel nostro metodo ci permette di generare segnali con componenti (che approssimano delle sinusoidi) con frequenze che non coincidono con nessuna di quelle delle armoniche della IFFT, si basa sulla seguente osservazione.

Consideriamo due onde sinusoidali di uguale ampiezza e frequenza (A, F_a) ma con differente fase:

$$a(t)=A \cdot \sin(2\pi \cdot F_a \cdot t + \varphi)$$

$$b(t)=A \cdot \sin(2\pi \cdot F_a \cdot t + \varphi + \alpha)$$

Consideriamo t che varia nell'intervallo $I=(0,T)$ e in questo intervallo eseguiamo la somma di due funzioni $a'(t)$ e $b'(t)$ ottenute da quelle di partenza come segue:

$$a'(t)=a(t) \cdot (1-t/T), \quad b'(t)=b(t) \cdot (t/T)$$

Si ottiene una funzione $c(t)$ (fig.1) che per valori di α piccoli approssima un segnale sinusoidale $d(t)$ di frequenza $F=F_a+\alpha/2\pi T$ (diversa da quella delle funzioni di partenza).

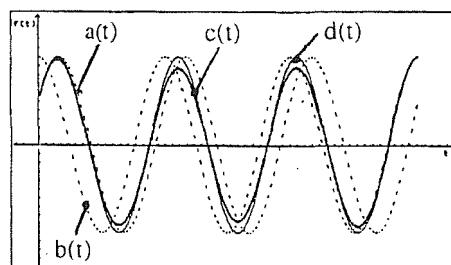


Fig.1.Esempio di sfasamento

Nell'algoritmo si sintesi, come vedremo in seguito, le funzioni $a(t)$ e $b(t)$ corrisponderanno a

due armoniche di uguale ordine di due diverse IFFT: i risultati di quest'ultime verranno sommati opportunamente in modo da ottenere segnali con componenti del tipo $c(t)$. In tab. 1 è riportato il SNR fra la sinusoide $d(t)$ e il segnale $d(t)-c(t)$, in funzione dello sfasamento α e di K , parametro definito come $2\pi/\alpha$.

Tabella 1

K	α (gradi)	SNR (dB)
2	180	2.8
4	90	13.4
8	45	25.1
16	22.5	37.0
32	11.25	49.1
64	5.63	61.1
128	2.81	73.2

Si considerano solo i valori positivi di α perché SNR è invariante rispetto al segno della fase. Con una serie di determinati artifici [4], si riesce ad avere un SNR>80dB anche con $\alpha=45^\circ$, come è mostrato nella tab. 2.

Tabella 2

K	α (gradi)	SNR (dB)
2	180	2.2
4	90	39.4
8	45	83.0
16	22.5	84.5
32	11.25	93.8
64	5.63	105.3
128	2.81	117.2

Sfruttando il meccanismo degli sfasamenti si possono generare componenti con frequenza qualsiasi e con risoluzioni di 0.2-0.3 hz, utilizzando opportune IFFT con un basso numero di armoniche e consentendo così di

ridurre il costo computazionale finale.

Algoritmo di sintesi

Poniamo di dover generare un segnale monofonico composto da M onde sinusoidali contemporanee. In ingresso avremo due vettori di M elementi; i j -esimi elementi di questi vettori sono la frequenza F e l'ampiezza A che l'onda j -esima deve avere nel frame che stiamo prendendo in considerazione.

La strutturazione a blocchi dell'algoritmo principale è la seguente :

```

while true do
begin
  for j:=1 to M do
  begin
    < INVILUPPO >
    < SFASAMENTO >
    < TRASFORMAZIONE >
    < ACCUMULO >
  end;
  < IFFT >
  < MIX FINALE >
end;

```

Descriviamo brevemente il funzionamento dei singoli blocchi. Le operazioni del ciclo più interno verranno effettuate M volte, cioè una per ogni onda sinusoidale, ed il tutto verrà ripetuto per ogni frame da creare:

<INVILUPPO>. Serve ad acquisire i valori di frequenza (F) e di ampiezza (A) da una prestabilita zona di memoria.

<SFASAMENTO>. Dato il valore di frequenza (F) e la frequenza base della IFFT (F_0), si ricava il numero dell'armonica a frequen-

za più vicina, $j=0.5+F/F_0$ e lo sfasamento necessario per approssimare quella data onda con $\alpha=2\pi T(F-jF_0)$.

<TRASFORMAZIONE>. Questo blocco esegue la trasformazione da coordinate polari nei corrispondenti numeri complessi, necessari alla IFFT.

<ACCUMULO>. In questo blocco vengono effettuati gli accumuli nei vettori che costituiscono l'input per le IFFT.

<IFFT>. Vengono utilizzate 2 IFFT reali (equivalenti ad una complessa) per ottenere ogni segnale monofonico: con la prima si considerano tutte le armoniche $a(t)$, mentre nella seconda quelle del tipo $b(t)$.

<MIX FINALE>. Vengono miscelati i vettori di numeri reali ottenuti dalle IFFT tramite delle tabelle di correzione opportune, in modo da ottenere la somma di tutte le onde di tipo $c(t)$.

Gli ultimi due blocchi devono essere eseguiti, in un'ipotetica implementazione in real-time, ad intervalli di T sec. esatti.

Simulazioni e valutazioni sul costo computazionale

Nella fig.2 della pag. successiva sono mostrati esempi di segnali ottenuti con il metodo degli sfasamenti, nella versione che tiene conto di tutte le ottimizzazioni: l'errore introdotto è a onda piena ed è amplificato 4000 volte per poter essere visibile. Vengono considerati 3 frame e l'intervallo visualizzato corri-

sponde ad un intervallo temporale pari a $4T$.

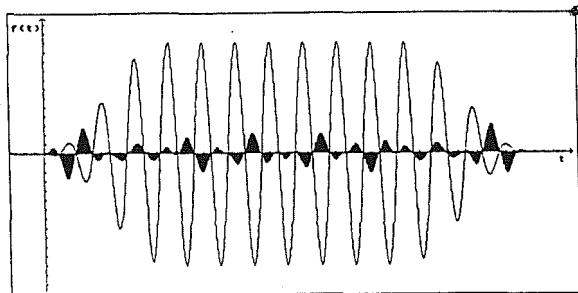
Il costo computazionale è stato valutato prendendo in considerazione le operazioni principali di moltiplicazione, addizione (o sottrazione) ed accesso in memoria, supponendo che i tempi di lettura e di scrittura in memoria siano uguali. Vengono trascurate le restanti operazioni, perché il loro numero è piccolo rispetto a quello delle operazioni principali. I risultati relativi ad un segnale stereo sono stati confrontati con quelli relativi al metodo dell'oscillatore digitale a lettura tabellare e riportati nella tabella 2.

Risultati e conclusioni.

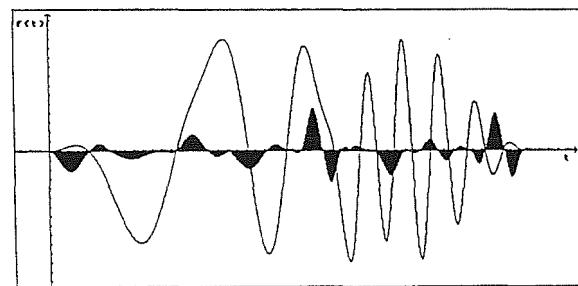
Al fine di verificare le potenzialità del metodo proposto, sono state fatte delle prove di sintesi (in tempo differito) con simulazione di FM, filtraggio, rumore bianco, campionamento e di segnali con spettri dinamici. Risultati particolarmente interessanti sono stati ottenuti nelle imitazioni della fase di rilascio di suoni, nei riverberi di strumenti quali l'organo a canne ed il saxofono e nell'imitazione del pianoforte.

In definitiva col metodo degli sfasamenti si riesce ad utilizzare la sintesi additiva per generare segnali con onde sinoidali a frequenza qualsiasi entro la risoluzione di 0.03 Hz, con un rapporto Segnale/Rumore superiore agli 80 dB, ad un costo computazionale che è, nel caso di qualche migliaio di componenti, inferiore ad un decimo rispetto a

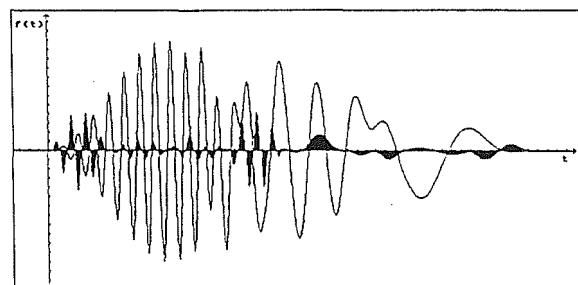
Aampiezza e frequenza costanti, SNR = 88.5 dB.



Aampiezza costante, frequenza variabile, SNR = 85.9 dB.



Aampiezza e frequenza variabili, SNR = 84.9 dB.



Due componenti con ampiezza e frequenza variabili, SNR = 84.9 dB.

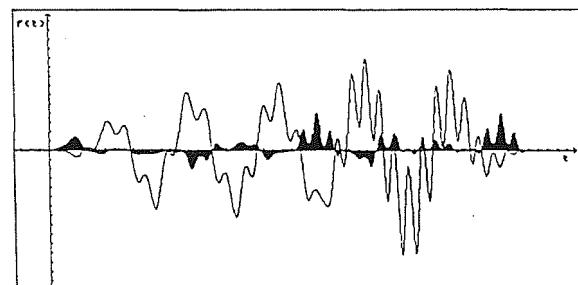


Fig. 2. Esempi di segnali ottenuti con il metodo degli sfasamenti.

quello richiesto dal metodo dell'oscillatore digitale.

Nella tabella 3 vengono riportati i valori realtivi ai due metodi: si

può notare come il metodo degli sfasamenti sia conveniente a partire da 250 oscillatori.

Tabella 3

M	Oscillatore Digitale			Metodo degli Sfasamenti		
	Molt.	Add	Mem.	Molt.	Add	Mem.
125	12	30	60	16	23	37
250	24	60	120	17	25	39
500	48	120	240	19	29	43
1000	96	240	480	22	37	50
2000	192	480	960	30	52	66
4000	384	960	1920	46	82	97
8000	768	1920	3840	78	142	158

Questo lavoro è stato svolto con il contributo del Prog. Finalizzato del CNR "Sistemi Informatici e Calcolo Parallelo" Sottoprog. 2.

Bibliografia

- [1] J.Snell:"*Design of a Digital Oscillator that Will Generate up to 256 Low-Distortion Sine Waves in Real-Time*", Computer Music Journal Vol.1 n.2, pp.4-25 (1977).
- [2] "The FDSS Studio, System software for Apple Mcintosh Computer", Computer Music Journal Vol.16, n.3, p. 99 (1988)
- [3] X. Rodet, Ph. Depalle: "A new additive synthesis method using inverse Fourier transform and spectral envelope" ICMC Proceedings (San Josè, CA, USA) pp. 410-411, 1992.
- [4] M.Barutti; "Un Metodo Efficiente per la Sintesi di Segnali Acustico-Musicali Mediante Trasformata Inversa di Fourier", Tesi di laurea, Corso di Scienze dell'Informazione, Univ. di Pisa, (1992-'93).

[5] G. De Poli:"*A Tutorial on Digital Sound Synthesis Techniques*" Computer Music Journal Vol 7, n.4, pp. 8-26 (1983).

[6] J.A.Moorer:"*Signal Processing Aspects of Computer Music : A Survey*", Proceedings of the IEEE, vol.65, n. 8 pp. 1108-1132 (1977) .

[7] G.Bertini, M.Chimenti, F.Denoth:"*TAU2: Un Terminale Audio per Esperimenti di Computer Music*", Alta Frequenza, Vol. XLVI n.12, pp.600-609, (1977).

[8] Cor Jansen:"*Sine Circuitu*", ICMC, Proceedings, (Montreal, CANADA), pp.223-225, (1991).

[9] M. Barutti, G. Bertini: "Valutazioni sul costo computazionale di una tecnica di sintesi basata su FFT inversa" Nota Int. IEI (in stampa).

[10] E.O.Brigham:"*The Fast Fourier Transform*", Prentice-Hall, (1974).

[12] H. Chamberlin:"*Musical Applications of Microprocessors*", Hayden, (1980).

ANALISI DELL'EFFETTO DEL TOCCO SUL TRANSITORIO D'ATTACCO DEI SUONI DI UN ORGANO A CANNE A TRASMISSIONE MECCANICA

Laura Bazzanella, Giovanni B. Debiasi

C.S.C. - D.E.I., Università di Padova, via Gradenigo, 6A - 35131 Padova

tel. +39 49 8287500-fax +39 49 8287699-Email debiasi@paola.dei.unipd.it

Ricerca svolta con il contributo di General Music S.p.A.-S.Giovanni in Marignano (FO)

ABSTRACT

In order to obtain an accurate electronic imitation of a pipe organ, it is necessary to pay attention to the characteristics of pipe organ sounds. Particularly, in this ambit, we consider the problem of the "touch". In the pipe organs with mechanical transmission, the organists notice an answer of the instrument connected with the type of solicitation, i.e. the so-called "touch", and in the description of the phenomenon they stress overall the key velocity. In order to explain this intuitive observation, we have considered a pipe organ with mechanical transmission (built by the organ builder Mascioni in Padua). We have considered the sounds of various stops of the pipe organ, using for all the notes two different ways of touch: slow and fast. The definition of these two categories has been effected by organists that have played the instrument; in some cases we have used a "mechanical finger". We have then studied the sounds with the STFT (Short Time Fourier Transform) algorithm and with our software useful for the analysis and representation of the spectral components of a sound, paying attention to the transient. The observation of the results has proved a real difference between the same note played with different "touch". The utilization of STFT permitted to notice a remarkable sensitivity of the second spectral component: with fast "touch" the amplitude reaches the stationary value with oscillations, while with slow "touch" we observe an exponential evolution. Using a proprietary software for the definition and representation of the timbre of a sound, we got the evolutions of the timbre and the complexity of the considered sounds, obtaining very different behaviours. The application of our method to different pipe organs will allow the acquisition of more informations on the effect of the "touch", in view of a subsequent application for the synthesis of more natural sounds with electronic instruments.

Introduzione

L'organo è uno strumento in cui il suono è prodotto da canne alimentate mediante aria inviata da un mantice e regolata per mezzo di tasti. I suoi componenti principali

sono, oltre alle canne, i somieri, i mantici, la consolle, le tastiere e la trasmissione.

Dai mantici, la cui funzione viene espletata negli organi moderni da elettroventilatori, l'aria giunge al somiere (una cassa sopra la quale sono montate le canne); essa poi passa alle canne attraverso speciali valvole la cui apertura è comandata dalla tastiera e dai comandi dei registri. Il collegamento tra i tasti e i comandi dei registri della consolle ed il somiere avviene attraverso la trasmissione.

Negli organi a trasmissione meccanica i tasti sono collegati alle valvole (ventilabri) attraverso un sistema di tiranti o di leve; gli altri tipi di trasmissione prevedono invece un collegamento fra tasto e valvola di tipo elettromagnetico od elettropneumatico.

La maggior parte degli organisti giudica che la trasmissione meccanica sia più idonea a fornire la possibilità di influire sull'attacco di ciascuna nota attraverso modalità di tocco differenti; in particolare è apprezzata la meccanica cosiddetta sospesa, nella quale i tasti sono tenuti in posizione di riposo dalla molla che tiene chiuso il ventilabro, al quale sono sospesi direttamente mediante un tirante.

Alcuni esprimono invece la convinzione che il transitorio di attacco sia assolutamente indipendente dal tipo di trasmissione e di tocco. Esistono dunque pareri discordi

(per valutare solo due tra i vari pareri contrapposti si vedano, per esempio, [1][2]).

La questione appare quindi ancora aperta, ed è sembrato perciò utile fornire un nuovo contributo per la sua soluzione. A questo scopo è stata intrapresa una serie di indagini sull'influenza del tocco in organi pregiati sia di costruzione moderna sia di epoca barocca.

Vengono qui presentati i risultati delle analisi effettuate sui suoni di un organo moderno di grande pregio a trasmissione meccanica, ottenuti con modalità di tocco diverse in modo da modificare, ove possibile, il transitorio di attacco dei suoni di tale organo variano il tocco.

Sono stati presi in considerazione tre diversi procedimenti di indagine sul segnale registrato e successivamente campionato: applicazione della STFT (Short-Time Fourier Transform); applicazione di un algoritmo di separazione fra parte armonica e parte inarmonica del suono; rappresentazione con diagramma polare o cartesiano del timbro.

2. Strumento e modalità di esecuzione

Lo strumento utilizzato per le registrazioni è l'organo Mascioni del Conservatorio "Cesare Pollini" di Padova, costruito nel 1979 ed installato nel 1980; si tratta di uno strumento a tre manuali a trasmissione meccanica, con 52

registri comandati elettricamente. I materiali usati per la costruzione delle canne sono quelli tradizionali, cioè lega di stagno per le canne labiali, rame per le ance, ottone per le trombe

Sono state effettuate registrazioni relative a diversi registri dell'organo, considerando per ogni nota due modalità di tocco: un tocco "lento" ed uno "veloce".

La definizione di queste due categorie è stata effettuata da organisti che hanno offerto la loro collaborazione per suonare lo strumento; per ottenere una velocità di battuta estremamente ridotta e costante (0.5 mm/sec), è stato utilizzato un dito meccanico.

Ogni nota è stata suonata singolarmente, con una durata di almeno 5 secondi.

Per quanto riguarda la scelta dei registri, si è preferito considerare la III tastiera, poiché per essa la trasmissione è a meccanica sospesa.

I registri analizzati sono: Principale 8', Bordone 8', Quintadena 16', Viola da Gamba 8'.

3. Modalità di acquisizione

Per l'acquisizione del segnale è stato utilizzato un microfono direzionale (cardioide) Sennheiser MD421 con apposito preamplificatore microfonico.

Il segnale dell'organo è stato registrato su un registratore tipo DAT, marca Denon, modello DTR 2000 con dinamica 90 dB e

risposta in frequenza lineare tra 2 Hz e 22 kHz.

Si è pure utilizzato un fonometro B&K 2603, con relativo microfono a condensatore, per la misura della Intensity Level (dB) dei suoni registrati.

Con un oscilloscopio digitale è stato possibile inoltre effettuare un controllo visivo dei rumori di fondo.

Vista la disposizione simmetrica delle canne relative ad ogni registro (note pari collocate tutte nella parte destra, note dispari tutte a sinistra), la posizione del microfono è stata fissata in due punti distinti (a 2.5 m dalle canne di facciata) a seconda che si registrasse una nota pari o una nota dispari.

Il suono registrato è stato poi suddiviso in transitorio d'attacco e parte stazionaria.

4. Analisi effettuate

Per effettuare le analisi è stato utilizzato il programma SPC90, sviluppato presso la General Music S.p.a., integrato da opportuni programmi per il tracciamento dell'evoluzione del timbro sul diagramma polare [3][4] o su diagramma cartesiano.

Si illustrano brevemente qui di seguito le principali funzioni dei programmi utilizzati.

4a. STFT(Short-Time Fourier Transform)

Se una serie temporale possiede caratteristiche spettrali temporali varianti, come ad esempio

un'armonica che trasla in frequenza, la sua rappresentazione in termini di trasformata di Fourier non ne fornisce informazione.

La Short-Time Fourier Transform invece consiste, in linea di principio, in una serie di trasformate di Fourier, ognuna relativa ad un particolare istante dell'intervallo di analisi (ad ogni istante considerato viene associata la trasformata di Fourier della porzione di segnale posta "attorno" a quell'istante); essa quindi permette di seguire l'evoluzione temporale dello spettro del segnale[5].

Le porzioni di segnale da analizzare si ottengono moltiplicando il segnale stesso per una finestra di analisi che viene traslata nel tempo di un passo costante.

Nel caso specifico, è stata utilizzata una finestra di Hamming di 256 o 512 punti traslata di volta in volta di 64 o 128 campioni rispettivamente.

In questo modo sono stati ricavati gli andamenti temporali di ampiezza e frequenza delle armoniche del segnale; in particolare si sono considerate le prime sei armoniche.

4b. Separazione fra parte armonica e parte inarmonica del suono

Dato lo spettro (medio) del segnale, viene costruito un filtro con risposta in frequenza data da una serie di finestre centrate sulle armoniche significative; dopo un

prodotto fra lo spettro del segnale e la risposta in frequenza del filtro, un'operazione di trasformata di Fourier inversa sul risultato permette di ricavare l'evoluzione temporale della parte armonica del suono.

La parte inarmonica, costituita soprattutto dal soffio della canna, viene poi calcolata per differenza.

4c. Rappresentazione grafica dell'evoluzione del timbro

E' stato utilizzato il già citato metodo per la valutazione del timbro degli organi a canne presentato al VIII Colloquio di Informatica Musicale [4] che permette, fra le altre cose, di rappresentare un suono in un diagramma polare, nel quale la distanza dal centro individua la complessità, mentre l'angolo rispetto all'asse verticale individua il "colorito" del suono in esame.

Per migliorare la visualizzazione durante il transitorio, è stato utilizzato anche un diagramma cartesiano, che riporta in ascisse l'angolo (colorito) ed in ordinate la complessità; inoltre, sempre per lo stesso motivo, è stata prevista la possibilità di selezionare (con una funzione di zoom) esclusivamente la regione del piano interessata dalla traiettoria.

5. Presentazione dei risultati

5a. STFT(Short Time Fourier Transform)

L'osservazione degli andamenti delle ampiezze delle prime sei

armoniche (vedi fig.1 e 2) evidenzia un comportamento diverso fra note uguali suonate con modalità di tocco differenti.

Tale diversità di comportamento è stata riscontrata, in modo pressoché uniforme, per le note analizzate dei vari registri presi in considerazione. Si è così potuto riscontrare che il valore a regime viene raggiunto dopo una sovraelongazione, anche notevole, dell'ampiezza della 2° armonica nel caso di esecuzione con tocco veloce; nel caso di tocco lento, l'andamento è di tipo esponenziale o comunque molto meno brusco.

Nel caso dei registri a canne tappate (Bordone 8' e Quintadena 16'), caratterizzati da spettri privi delle armoniche di ordine pari, lo stesso comportamento si riscontra per la 3° armonica.

Questa caratteristica del comportamento in transitorio conferma pienamente quanto già enunciato da vari autori [6][7][8], mentre contraddice quanto affermato in [1].

5b. Separazione di parte armonica e parte inarmonica

Anche le evoluzioni temporali della parte inarmonica (soffio) nelle due condizioni di tocco risultano estremamente diverse (vedi fig. 3c e 4c). Anche in questo caso si sono riscontrati risultati complessivamente omogenei per le varie note esaminate.

La conclusione che si può trarre in questo caso è che, con il tocco

veloce, l'emissione iniziale di rumore (soffio) risulta più accentuata.

Tale conclusione fornisce una ulteriore conferma dell'effettiva influenza del tocco sul transitorio iniziale dei suoni delle canne; questo tipo di conferma non sembra sia stato preso in considerazione da altri autori.

5c. Rappresentazione grafica del timbro

Considerando l'evoluzione del timbro durante il transitorio d'attacco (fig. 5,6,7 e 8), si notano traiettorie differenti.

I diversi punti delle traiettorie corrispondono a intervalli temporali di 20 ms.

L'orientamento delle traiettorie, molto evidente sul diagramma cartesiano espanso, consente di effettuare una valutazione qualitativa di come si modificano la complessità e l'angolo relativo al timbro nel transitorio di attacco. Nella figura 6 si può notare che il timbro tende ad evolvere da valori di complessità ed angolo maggiori a valori minori.

Anche questa è una caratteristica abbastanza uniforme, già riscontrata anche in precedenti analisi [9].

L'influenza del tocco, in questo caso, consiste nel rendere differente questa modalità di evoluzione, passando dalla modalità di tocco lento a quella di tocco veloce.

Anche tutto ciò fornisce un'ulteriore conferma della sensibilità al tocco dello strumento, con un approccio

del tutto nuovo, mai preso sinora in considerazione per la mancanza del metodo adatto ad evidenziare l'evoluzione globale del timbro in transitorio.

6. Conclusioni

Il desiderio di approfondire sempre più accuratamente la conoscenza del comportamento degli organi a canne, da una parte, e l'esigenza di rendere sempre più realistico il suono degli organi liturgici elettronici, dall'altra, conducono ad analizzare in modo sempre più minuzioso le caratteristiche degli organi tradizionali.

In particolare, qui si è indagato sulla possibilità che il tocco influisca sul transitorio d'attacco dei suoni di tali strumenti.

I risultati delle sperimentazioni effettuate permettono di confermare che, negli organi a canne a trasmissione meccanica, diverse modalità di esecuzione delle note determinano diverse caratteristiche del loro transitorio d'attacco.

Tra le modalità di indagine adottate, due sono da considerarsi innovative e anche queste concordano pienamente con i risultati ottenuti mediante gli altri metodi di analisi già utilizzati da diversi autori, consentendo di poter affermare con ancor maggiore sicurezza che, effettivamente, il tocco influisce sul transitorio di attacco dei suoni degli organi a canne a trasmissione meccanica.

Bibliografia

- [1] E. Girardi, "Problematiche Relative all'Esecuzione su Organi Liturgici in Situazioni e Modalità Diverse di Alimentazione", III Convegno di Organologia sul Tema La Riforma dell'Organo Italiano, Pisa 1990.
- [2] P. Barbieri, "La Controriforma dell'Organo in Italia: Meccanica e Fonica dal 1930 al 1990", III Convegno di Organologia sul Tema La Riforma dell'Organo Italiano, Pisa 1990.
- [3] M. Dal Sasso, G.B. Debiasi, "Metodo per la Valutazione Automatica dei Timbri di Organi a Canne", VIII Colloquio di Informatica Musicale, Cagliari 1989.
- [4] G.B. Debiasi, G. Spagiari, "Metodi di Analisi delle Microvariazioni di Ampiezza e Frequenza dei Suoni Musicali e loro Applicazioni allo Studio di un Organo Barocco", IX Colloquio di Informatica Musicale, Genova 1991.
- [5] J.S. Lim, A.V. Oppenheim, "Advanced Topics In Signal Processing", Prentice Hall, 1988.
- [6] N.H. Fletcher, "Transients in the Speech of Organ Flue Pipes-A Theoretical Study", ACUSTICA, vol.34, pp.224-233, 1976.
- [7] T.L. Finch, A.W. Nolle, "Pressure Wave Reflections in an Organ Note Channel", Journal of the Acoustical Society of America, vol.LXXIX-5, pp.1584-1591, 1986.
- [8] S. Caddy, H.F. Pollard, "Transients Sounds in Organ Pipes", ACUSTICA, vol.8, pp.277-280, 1957.
- [9] M. Dal Sasso, G.B. Debiasi, G. Spagiari, "Method for Automatic Evaluation of Timbre and Fluctuations of Pipe Organ Sounds", International Computer Music Conference, Montreal 1991.

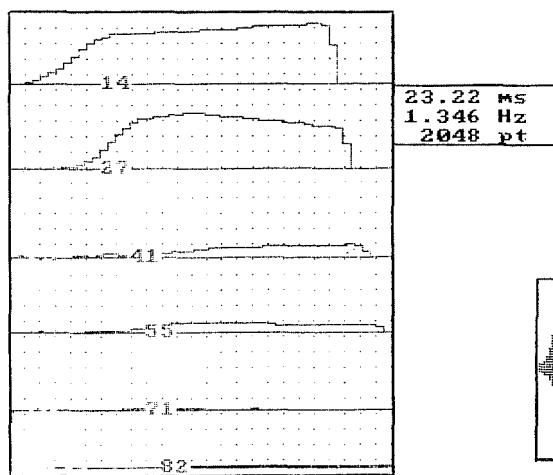


Fig.1 Andamento temporale dell' ampiezza delle prime sei armoniche - Nota RE2 - Registro PRINCIPALE 8' - Tocco LENTO

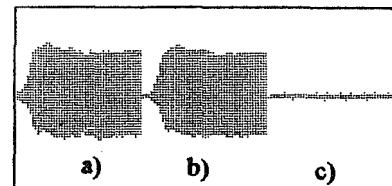


Fig.3 Evoluzione temporale del transitorio di attacco - Nota RE2 - Registro PRINCIPALE 8' - Tocco LENTO
 a) suono complessivo
 b) parte armonica
 c) parte inarmonica

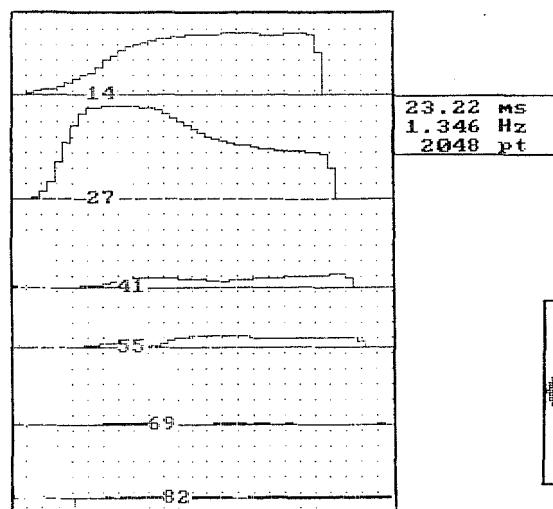


Fig.2 Andamento temporale dell' ampiezza delle prime sei armoniche - Nota RE2 - Registro PRINCIPALE 8' - Tocco VELOCE

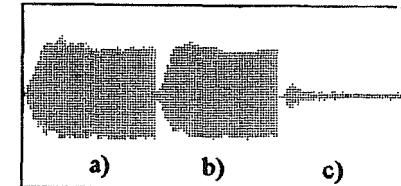


Fig.4 Evoluzione temporale del transitorio di attacco - Nota RE2 - Registro PRINCIPALE 8' - Tocco VELOCE
 a) suono complessivo
 b) parte armonica
 c) parte inarmonica

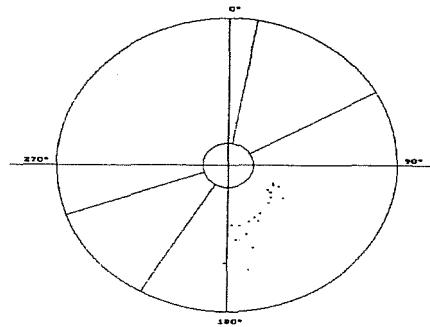


Fig.5 Diagramma polare dell'evoluzione del timbro - Nota RE2 - Registro Principale 8' - Tocco LENTO

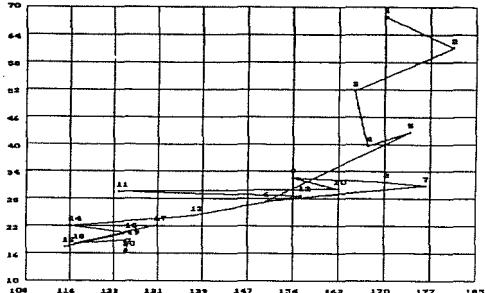


Fig.6 Diagramma cartesiano espanso (zoom) dell'evoluzione del timbro - Nota RE2 - Registro PRINCIPALE 8' - Tocco LENTO

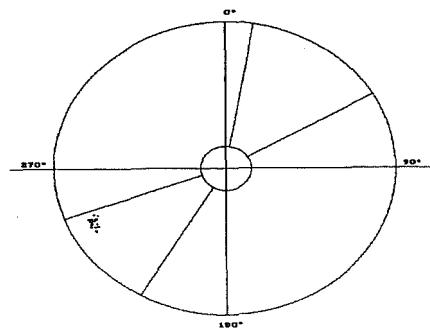


Fig. 7 Diagramma polare dell'evoluzione del timbro - Nota RE2 - Registro Principale 8' - Tocco VELOCE

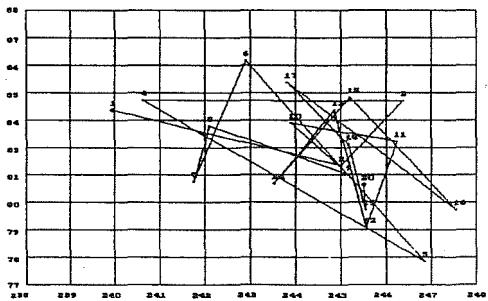


Fig.8 Diagramma cartesiano espanso (zoom) dell'evoluzione del timbro - Nota RE2 - Registro PRINCIPALE 8' - Tocco VELOCE

SOUND ANALYSIS METHODS BASED ON CHAOS THEORY

Angelo Bernardi, Gian-Paolo Bugna, Giovanni De Poli

CSC-DEI, Universita' di Padova, Via Gradenigo 6a, 35131 Padova PD, Italy
email: depoli@dei.unipd.it

Abstract

Musical signals presents several aspects that are not detected by classical analysis methods, such as time frequency analysis techniques. For instance in signals produced by real instruments, the non linear dynamics of the excitor may often result in some small or large degree of turbulence during the evolution of the sound, or in the production of non periodic sound (such as multiphonics tones). We investigate the possibility of using analysis methods based on chaos theory both for studying significant properties of the signal and of the production mechanism.

1. MUSIC SIGNAL AS FRACTAL (CHAOTIC) SIGNAL

Chaotic signals are of particular interest and importance in experimental physics because of the wide range of physical processes that apparently give rise to chaotic behaviour. In particular the physical systems interested by this kind of signals are characterised by strong non-linearity and a fractal dynamic nature. From the point of view of

signal processing, the detection, analysis and characterisation of signal of this type present a significant challenge and an opportunity to explore and develop completely new classes of algorithms for signal processing. There is the experimental evidences that some music signals show a fractal nature, a first example are the musical sound produced by a non-linear dynamic excitor with chaotic behaviour [1]. Besides it's shown that musical sound presents a pitch fluctuation (turbulence) that can be modelled and quantified by fractals. Spectral density measurements of the pitch variation in various types of music show their common correlation with the fractal world. More over some kind of musical computer aided composition are tied to the self similarity property of generator of event sequence. All the above theoretical considerations and the experimental evidence drive us to use fractals as a mathematical model for the study of music signals. The music signal waveform is studied by new analysis techniques based on the concepts of fractal geometry.

One of the main purpose of these

kind of approach is to determine whether the structure of a musical signal can be considered self-similar, and hence it is possible to quantify the degree of turbulence of the signal itself as reflected in the fragmentation of its time graph. The most important idea we focus on is the fractal dimension of sound signals because it can quantify their graph's fragmentation. Since the relationship between turbulence and its fractal dimension or the fractal dimension of the resulting time graph is little understood, in this paper we conceptually equate the amount of turbulence in a musical sound with its fractal dimension. To compute it, is used an algorithm based on morphological filters that iteratively expand and contract the signal's graph. Thus it is possible to determine the most significant properties of music signal if the starting hypothesis of fractal signal is sufficiently verified. Besides with this new analysis method based on fractal geometry it is possible to obtain useful information for sound characterisation also when the music signals is not closely related with a fractal.

2. FRACTAL SIGNAL MODEL AND LOCAL FRACTAL DIMENSION

To understand how fractal geometry, chaotic signals are related with musical signal we can

examine the basic concepts of exact self similarity and statistical self similarity. An example of exact self similarity is the Von Koch curve; in fact each small portion of the curve, when magnified, can reproduce exactly a larger portion. The curve is said to be invariant under the changes of scale. An actual object rarely exhibit exact self similar at every scale factor; however they often possess a related property of statistical self similarity, where a small portion of the curve looks like (but not exactly like) a larger portion. Formally we can say that a signal is statistical self similar if the stochastic description of the curve is invariant under scale change.

Fractal sets are characterised by statistical self-similarity properties; objects obviously remain different in details related with scale change. The way, in which the detail varies as one changes scale, can be characterised by a parameter called fractal dimension.

From the application of signal theory it is possible verify that there is a direct relationship between the fractal dimension and the logarithmic slope of the spectral density of a fractal signal [2]. For instance a one dimensional Fractional Brownian Motion signal with a fractal dimension D is characterised by a spectral density proportional to $1/f^\beta$ where $\beta=(5-2D)$.

However the statistical self

similarity above considered extend from arbitrary large to arbitrary small scales. For actual signal this property can be formulated only over a finite range of scale changes. Therefore in the case of real signal the hypothesis of a statistical self similarity can be sustained only over a finite range of time scale. For instance the sound have a finite duration and in the discrete time they have also a finite bandwidth. However in this range the fractal dimension should appear constant, but on the other hand for real signals the hypothesis of self similarity is only approximately verified and then the fractal dimension results a function of the time scale factor ε too. Moreover for music signal that can be described by non stationary process the characteristic of the signal varies in function of time: hence also the fractal dimension will reflect this property.

The estimation of a global fractal dimension will give an average value of this variation of little interest for a sound characterisation. Hence, instead of a global dimension we estimate a local fractal dimension (LFD) computed on a window covering portion of a musical signal. Since the music signal generally is non stationary, the measurement of different windows will not be same. If we move the window along the time evolution of the signal we can underline various

behaviours in different portions of the signal characterised by different fractal dimension.

The local fractal dimension results variable as a function of time and of scale factor $LFD(t, \varepsilon)$. We can note the similarity with Short Time Fourier Transform (STFT) defining local time/frequency variations of the signal, and remember the relationship between fractal dimension and spectral slope.

Numerous algorithms exist to estimate fractal dimension. We employed an efficient algorithm, recently developed by Maragos [3] [4], based on morphological filtering of time graph of the signal to compute variation of the cover areas vs. time scale. This algorithm allows to estimate $LFD(\varepsilon)$ which for each ε is equal to the slope of a line fitted to the log-log plot of the cover area over a moving window. We employed a window of 10 scales $\{\varepsilon, \varepsilon+1, \dots, \varepsilon+9\}$.

3. FRACTAL DIMENSION AS SCALE FACTOR FUNCTION

Here we report some fractal dimension analysis on real musical signals, such as multiphonics sounds on woodwinds (clarinet and oboe) and wolf-note on double-bass together with conventional sounds on the same instruments.

Fig. 1.a and 1.b show time behaviour of a clarinet sound and its fractal dimension vs. scale factor. It can be seen that fractal dimension analysis of a real musical signal shows interesting properties. Particularly we can observe that many referred signals show fractal properties such as the invariance of

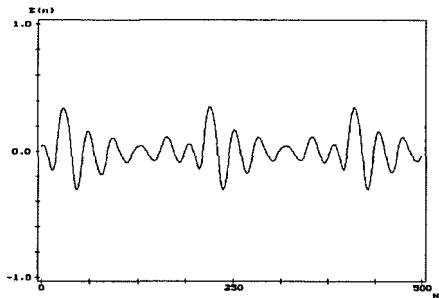


Fig. 1.a

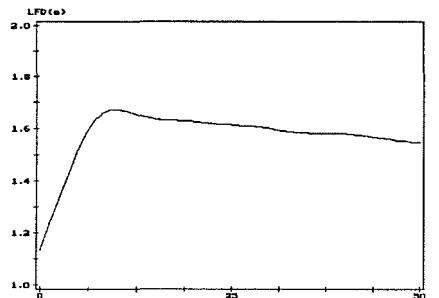


Fig. 1.b

local fractal dimension vs. scale factor.

We can see that fractal dimension isn't representative of the analysed instruments too: indeed the same instrument can generate sounds with different fractal dimension.

After a great number of tests we established that we obtain best results

using time windows of 1 or 2 periods of the examined signals. Indeed with a largest number of periods we can reach a saturation condition in which to a scale factor variation don't concur an adequate variation of the cover area. However to avoid this problem we can use a higher sampling frequency.

We can observe that fractal analysis is not sensitive to a magnitude (attenuation) factor applied to the musical signal too. Indeed fractal dimension is estimated with a straight line slope on a logarithmic scale: so the possible magnitude/attenuation factor reflect back only on intercept of the straight line interpolation, not its slope. To obtain best results we suggest to use full range signal amplitude; this way we have a less sensitive dependence of the estimated dimension from quantization errors.

In the previous example we applied a fractal model to a substantially quasi-periodic signal. To better characterise turbulence we prefer to separate this component from the quasi-periodic part of the signal. We extract the frequency and amplitude deviations of the harmonics by STFT analysis and apply the fractal model. Fig.2.b shows the estimated fractal dimension of amplitude deviations (fig.2.a) of the first C4 partial played in the principal register with

a pipe-organ. These fluctuations are well approximated with fractal dimension values 1.6. Successive partials show similar behaviours. These evidences reveal that fractal

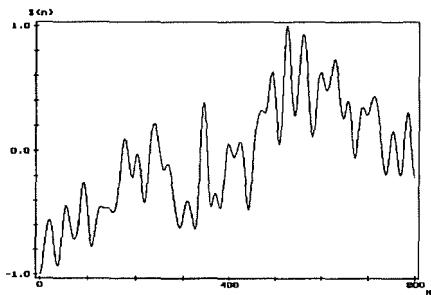


Fig. 2.a

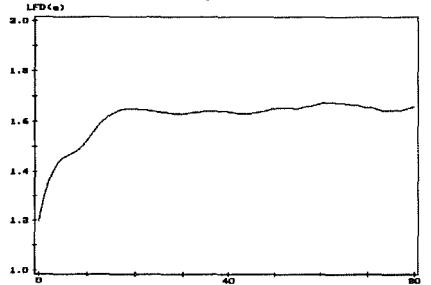


Fig. 2.b

modelling in control signals is widely justified.

A great number of the signals we analysed keep a sufficiently uniform local fractal dimension varying the scale factor ϵ . Certain signal don't verify this condition anyway. Some seem to stabilise very slowly on a steady fractal dimension. Others never exhibit a stable fractal dimension, reed excited registers for instance. Such musical signals are probably characterised by an absence of turbulence, so the primary conjecture of the fractal modelling is not

verified.

Information deduced with the previous analysis could be employed in a sound synthesis model. We can think to an additive synthesis model in which we can rule partials' amplitude and frequency fluctuations by control signals consistent with fractal analysis. In simpler cases it should be enough to generate a control signal with a constant local fractal dimension varying the scale factor. Fractal interpolation can be used when we desire to specify a general behaviour too.

Synthesis techniques could presumably be applied for signals with a local fractal dimension varying (vs. scale factor) according an assigned function of scale factor. This way it should be possible to look at sound synthesis for signals where the invariance of the local fractal dimension vs. scale factor of control signals is not verified.

4. FRACTAL DIMENSION AS FUNCTION OF TIME

Computing variations of the fractal dimension as function of time for a musical signal may show the time varying turbulence of the various phases of the sound.

To remark this information we choose the value of fractal dimension corresponding to a fixed scale factor, so we calculate this value for different windows,

namely different portions of the same signal in subsequent time. The choice of the designated scale factor is settled from two prominent factors: a choice of a too low factor ϵ could give rise to a wrong value of the dimensions, while a large scale factor imply a long computation time. Due to this factors we estimate that the best reference value is in order of tens.

As a result to the dimension estimation for that particular scale factor on consecutive signal sections, it is possible to plot a graph reporting the estimate dimension vs. time.

Results of this analysis are shown in the following picture (Fig. 3), in which we report time behaviour of the signal and the corresponding fractal dimension variations. In this picture we used observation windows of 1,000 samples, scale factor was fixed to $\epsilon=10$ while the estimation of straight line slope (dimension) was discerned on ten points.

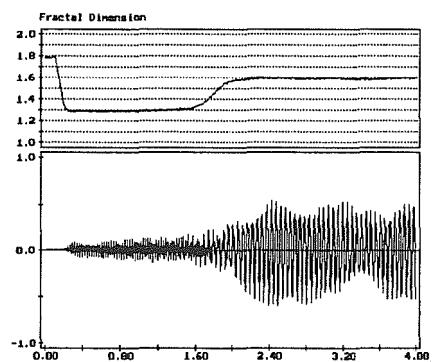


Fig. 3

After a transient where fluctuations may be perceptible, generally the fractal dimension of a real instrument sound shows a tendency to settle on a constant value. We made several tests on the same signal, varying the number of the samples of the window used for the dimension computation. Using windows with a lower number of samples the behaviour is similar to that shown in the reported pictures but it marked out large fluctuations on the average value. Clearly a reduction of the number of samples involves a diminishing of the time required for computation.

Nevertheless previous pictures clearly show fluctuations of the fractal dimension into the attach (and decay) transient for sounds radiated by real instruments. Of course such information could be useful applied into synthesis models so to evoke a large naturalness of the artificial sounds.

5. GENERATOR'S DYNAMIC ANALYSIS IN STEADY SOUNDS

Besides fractal modelling of signals, chaos theory developed tools for the analysis of the sound generation mechanism. Namely, it is possible to resolve the particular behaviour originating the sound. Particularly this approach is useful in musical acoustics in steady state behaviours.

Reconstruction of the attractor in time-delayed phase space [5] is the most important tool, associated with Poincaré map technique, that allowed a reduction of one dimension for system description. These methods complement classical time-frequency analysis techniques, and can show many aspects that aren't detected by a simple FFT. In particular we investigate the regime produced by some self-sustained musical instruments, with the aim of exploring the physical phenomena underlying sound production mechanism. Multiphonics of oboe, clarinet and recorder and wolf-note of contrabass are analysed. In this way the experimental evidence for

chaos in multiphonics tones derived from measurement of their power spectra can be confirmed by the reconstructed attractors in the phase space of the dynamic system. Fig. 4 shows a 3D phase space reconstruction and the concerning Poincaré map for a clarinet multiphonic. The attractor is chaotic (biperiodic).

We can evaluate the fractal dimension of the attractor reconstructed in phase space: this parameter is quite different from the fractal dimension of the signal (vs. time) graph we saw. The dimension of attractors can be evaluated by embedding the time series in a higher space. The

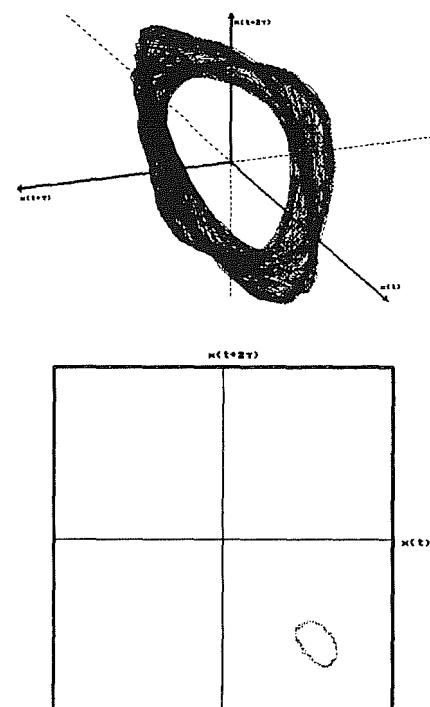


Fig. 4

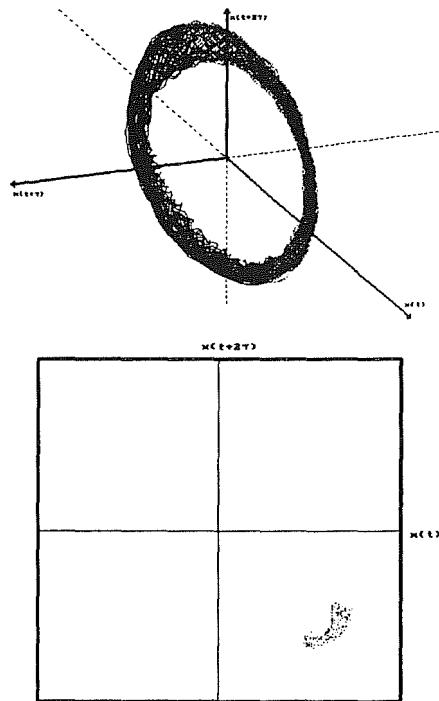


Fig 5

correlation dimension D [6] is measured by embedding a single measured time series in a higher-dimensional space, so to reconstruct the phase space of the dynamic system. In particular the estimated dimensions of the attractors can show that sometimes the sound has the characteristics of chaotic dynamics with a fractal (non integer) dimension, and other times it shows a behaviour with a biperiodic spectrum. Thus the specific typology of the reconstructed attractors shows that self-sustained musical instruments can be modelled by non-linear dynamic systems with a low degree of freedom. In fact a phase-locked biperiodic spectrum is characteristics

of low-dimensional chaotic attractors in a mechanical or fluidodynamic systems, in particular it seems that the real musical instruments analysed show a behaviour like a quasi-periodic route to chaos.

So we examined the multiphonic sound of fig. 4 with a dimension 2.01, suggesting a biperiodic dynamic. Instead for the clarinet multiphonic #33 we detected three subsequent kind of steady state behaviours, with attractor dimensions 1.23 (like a periodic and a little bit noised sound), 3.24 and 3.82, suggesting a quasi-periodic route to chaos (see respectively fig. 5-6-7). Only the first regime is quite different from the others: frequency

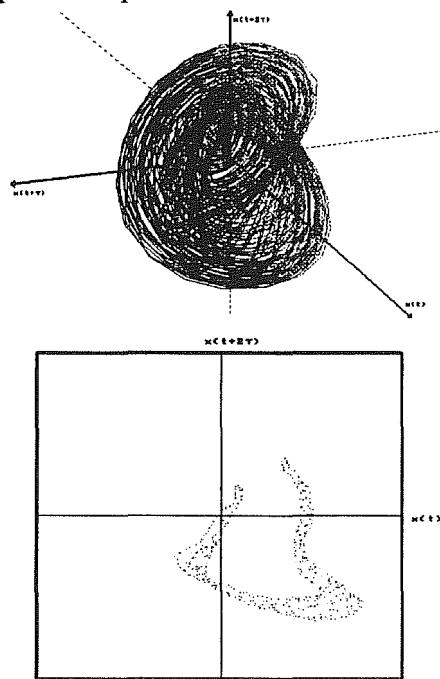


Fig. 6

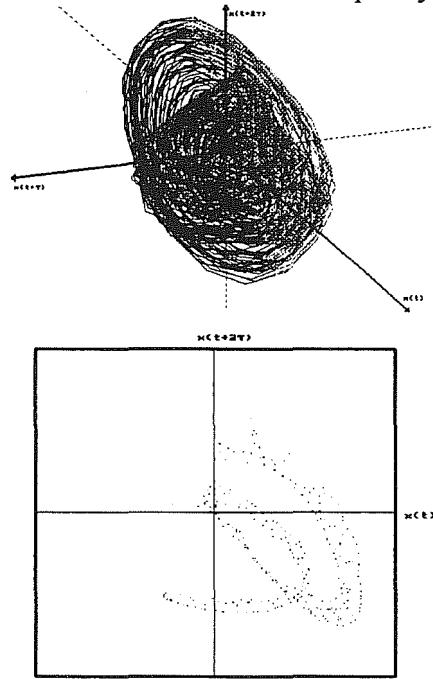


Fig. 7

analysis appears similar in the second and third case, and hearing too. Reconstruction delay choice in phase space is a very critical factor for an expressive description of phenomena. For musical signals we obtained good results picking out a zero (generally the first) of the self-correlation function.

Bifurcation diagram is another useful tool for chaotic system analysis: it shows the proceeding of the steady state behaviour varying a system parameter. This tool helps the correct choice of parameters in the simulation of physical models. Often it reveals the particular route to chaos of the model (often we find that it's different from the route to chaos of the real instrument). Thus a basic shortcoming of many physical models for sound synthesis is pointed out. In fact these models

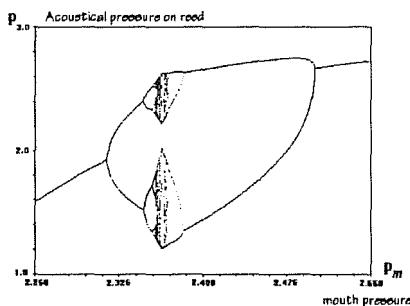


Fig.8

does not allow to obtain the global behaviour (e.g. route to chaos) of a real instrument. For example, fig. 8 shows a bifurcation diagram for the classic physical model of the clarinet [7] exhibiting a non-

realistic period-doubling route to chaos.

6. REFERENCES

- [1] D.H. Keefe, B. Laden: "Correlation dimension of woodwind multiphonic tones", J.Acoust. Soc.Am., vol.90, n.4, pp.1754-1765, 1991.
- [2] R.F. Voss: "Random fractal forgeries", in *Foundament algorithms for computer graphics*, R.A.Earnshaw ed., Springer-Verlag 1985.
- [3] P.Maragos: "Fractal aspects of speech signals:dimension and interpolation", Proc. ICASSP 91, 417-420, 1991.
- [4] P.Maragos, Fang-Kuo Sun: "Measuring the fractal dimension of Signals: Morphological Covers and Iterative Optimization", IEEE Transaction on Signal Processing, 1993.
- [5] W. Lauterborn, U. Parlitz: "Methods of chaos physics and their application to acoustics", J.Acoust.Soc.Am., vol.84, n.6, pp.1975-1993, December 1988.
- [6] P. Grassberger, I. Procaccia: "Measuring the strangeness of strange attractors", Physica, vol.9D, pp.189-208, 1983.
- [7] A. Bernardi, G.P. Bugna, G. De Poli: "Analysis of musical signal with chaos theory", Proc. Int. Work. Models and Representations of Musical Signal, Capri 1992

COUNTERWAVE

A program for controlling degrees of independence between simultaneously changing waveforms

Dr. Arun Chandra
Institute of Applied Arts
National Chiao Tung University
Hsinchu, Taiwan 30050
Republic of China
fax: +886-35-712-332
email: arunc@cc.nctu.edu.tw

Introduction

In the 1950s, W. Ross Ashby published his book *An Introduction to Cybernetics*, which presented structural transformations in a way that allows their description to be independent from their medium of implementation. At that time, the science of cybernetics, started by the work of mathematician Norbert Wiener and engineer Claude E. Shannon, generated an enormous amount of interest due to the possibilities it offered with regard to devising a language for the analysis of systems that was independent of any one field. People from many disciplines were attracted to it: anthropologist Margaret Mead, biologist Heinz von Foerster, mathematician John Neumann, psychologist Gregory Bateson, and many others, all took part in the "Macy"

conferences in the early 50s, and the subsequent founding of the American Society for Cybernetics.

Also in the 1950s, Lejaren A. Hiller and Leonard Isaacson programmed a computer (the ILLIAC II) to generate output based on applications of traditional contrapuntal rules. This output was transcribed into music notation, and performed by a string quartet, and was the first composition created with the assistance of a computer.

In the 1970s, the composer Herbert Brün (who had been involved with compositional experimentation with technology since his work in the 1950s at the Cologne Radio Studios in West Germany), began work at the University of Illinois on his project SAWDUST, with the initial assistance of Gary Grossman, and later Jody Kravitz and Keith Johnson. SAWDUST, originally written for a

VAX 11/780 and then ported to a 386-PC by Johnson, took its synthesis paradigm not from the mathematical models of Fourier synthesis, but the transformational ideas of cybernetics. SAWDUST allows for the specification of square waves, which are then subject to linear transformation from a specified initial to a specified final state. The program currently has five transformational algorithms: *vary*, *turn*, *merge*, *mingle*, and *link*.

My work on CounterWave began by wondering what would happen if the idea of counterpoint were applied, not to notes, but to states of waveforms?

CounterWave

Using CounterWave requires two steps: [1.] Specification of the waveforms and their changes. [2.] Specification of the relationships between waveforms.

Definitions

Relationship: The degree of consequence the presence or absence of one waveform has on the state of another.

Waveform: An iterated sequence of states, where a state may or may not change upon iteration, depending on its constituent segments.

State: A specified sequence of segments. A state can be made of any combination of segment types. A segment can occur more than once within a state.

Segment: A specified sequence of samples. As of this writing, a segment can be one of three types:

1. A wiggle: All samples have the same amplitude.
2. A twiggle: The samples have amplitudes that rise to a potentially fluctuating “peak,” then return to their starting value. Both rise and fall are linear.
3. A ciggle: The amplitudes rise to and fall from a potentially fluctuating “peak,” following the paths of two second-order polynomials.

All three types can be “slanted,” that is, a segment’s starting amplitude is the previous segment’s ending amplitude. The segment’s specified amplitude is taken as its ending amplitude. If a slanted segment is the first member of a state (*i.e.*, if there is no “previous segment”), its starting amplitude is assumed to be zero.

Change

Each segment type has either two or four variables, described below. Every variable is given an initial value, minimum and maximum limits, and a rate of change. Upon each iteration of a state:

- If a variable has a non-zero rate of change, its rate is added to or subtracted from its current magnitude.
- If a variable has a rate of change that is zero, it repeats its magnitude.
- If a variable reaches its minimum or maximum limit, it changes the direction of its

changing, *i.e.*, if it was growing, it will start shrinking, and *vice-versa*.

Thus, every variable that has a non-zero rate of change has a cycle length of

$$\text{cycles} = \frac{2(\max - \min)}{\text{rate}}$$

After the state has iterated *cycles* number of times, the variable will return to its initial magnitude.

Since a segment can have either two or four variables, and each variable can have a unique cycle length, the return of a variable to its initial magnitude does not necessarily result in the return of a segment to its initial configuration.

A wiggle

A wiggle has two variables:
[1.] Length in samples (0–1000).
[2.] Height (amplitude) (± 32767).

Below is an example of a data file for creating seven segments that are wiggles.

The first line gives a unique identifier for the segment, followed by its type. The second line gives, from left to right, the initial length of the segment, its maximum and minimum limits, and its rate of change. The third line gives the initial height (amplitude), its maximum and minimum limits, and its rate of change.

```
#init max min rate
w0 wiggle
100 100 20 4.5
10000 20000 -20000 10

w1 wiggle
20 130 20 4.5
```

```
-10000 10000 -10000 20
w2 wiggle
50 100 20 4.5
10000 20000 -20000 30

w3 wiggle
1 130 20 4.5
-10000 10000 -10000 40

w4 wiggle
1 100 20 4.5
10000 20000 -20000 50

w5 wiggle
1 130 20 4.5
-10000 10000 -10000 60

w6 wiggle
1 100 20 4.5
10000 20000 -20000 70
```

These seven segments are combined to form a state. The sequence of segments in this state is: w0, w1, w2, w3, w4, w5, w6.

Plots of this state at its 100th and the 700th iterations, followed by their respective FFTs, are displayed below (figures 1-4). In addition to the change in the amplitude configuration, please notice the change in length of the state.

The next example (Figures 5 and 6) uses the same segments as above, but four of the seven segments are slanted. The iteration number for each state is the same as that of Figures 1 and 2.

```
#init max min rate
w0 wiggle
100 100 20 4.5
10000 20000 -20000 10

w1 wiggle
20 130 20 4.5
-10000 10000 -10000 20

w2 wiggle slanted
50 100 20 4.5
10000 20000 -20000 30

w3 wiggle slanted
1 130 20 4.5
-10000 10000 -10000 40

w4 wiggle slanted
```

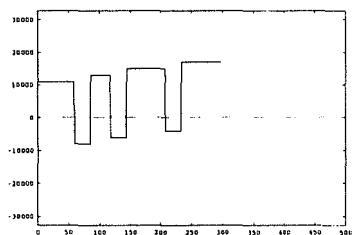


Figure 1: 7 segments, wiggles, 100th iteration

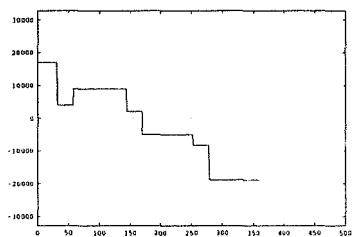


Figure 2: same as figure 1, 700th iteration

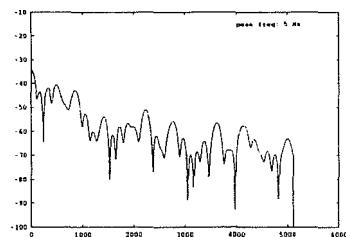


Figure 3: FFT of figure 1

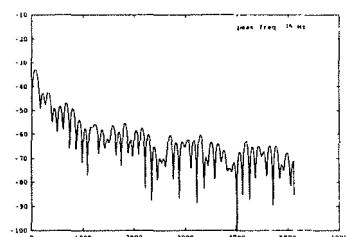


Figure 4: FFT of figure 2

```

1      100    20    4.5
10000 20000 -20000 50

w5      wiggle slanted
1      130    20    4.5
-10000 10000 -10000 60

w6      wiggle
1      100    20    4.5
10000 20000 -20000 70

```

A twiggle

A twiggle is a triangular wiggle. The idea behind a twiggle was to have a “triangle” in which all three sides could be in flux. Thus, there are four variables: [1.] Base length in samples (0–1000). [2.] Base height (amplitude) (± 32767). [3.] Peak height (amplitude) (± 32767). [4.] Peak location relative to the length of the base (0–1).

Figures 7 and 8 show a state that has 15 segments, all twiggles. Please notice that, unlike the wiggles example above, the length of the state in iterations 300 and 900 is approximately the same, although the waveform has substantially changed.

A ciggle

A ciggle is a twiggle with curved sides. Whereas the twiggle connected the base of its triangle to its peak with straight lines, a ciggle connects them with curved lines. A ciggle uses the same set of variables as a twiggle.

In Figures 9 and 10, four ciggles (c0—c3) are used to construct a state that has eight segments in the following sequence: c0 c1 c2 c3 c1 c0 c2 c1.

Coordination

Each waveform records aspects of its current state in a “window” variable, accessible by all other waveforms in formation. This window variable contains:

1. The number of segments in the state and their type.
2. The current maximum and minimum amplitudes of the entire state.
3. The current length in samples of the entire state (the sum of all the segments).

Upon beginning construction of a new state, each waveform checks the windows of all other waveforms in creation. After reading this information, the waveform can either ignore it, or act on it. If ignored, the waveform continues its configuration of changes. If not ignored, the waveform can:

1. Change the minima and maxima for some or all its variables.
2. Change the rate of change for some or all its variables.
3. Change some or all segments from being “slanted” to “straight,” or vice-versa.
4. Set the number of iterations for which it will ignore other waveforms.

The criteria used to decide whether to change a waveform’s variables can be relatively simple, such as “if x other waveforms are present, change some variables,” or variations on it. The criteria can also be more affectionate:

Jealousy: if another waveform’s dynamic range is greater, match its dynamic range.

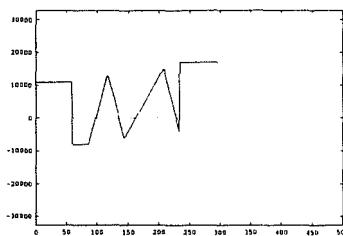


Figure 5: 7 segments, wiggles, some slanted, 100th iteration

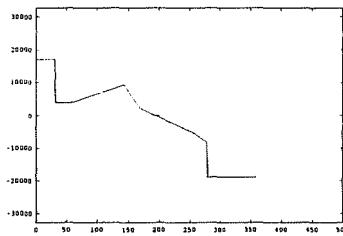


Figure 6: same as figure 5, 700th iteration

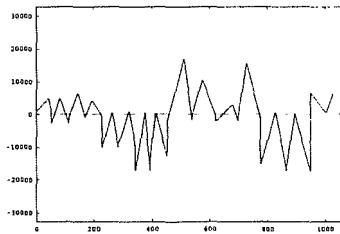


Figure 7: 15 segments, twiggles, 300th iteration

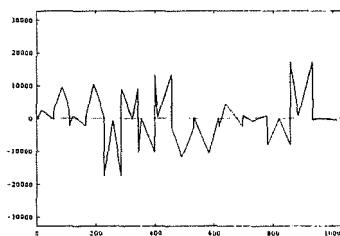


Figure 8: same as figure 7, 900th iteration

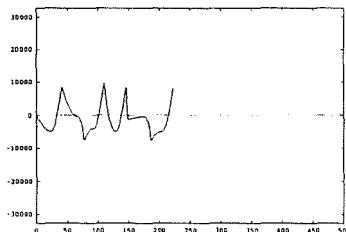


Figure 9: 8 segments, ciggles, slanted, 200th iteration

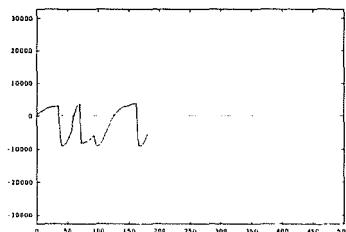


Figure 10: same as figure 9, 600th iteration

Revulsion: if another waveform's limits are within x , change all limits by a factor of y .

Me-too: if another waveform's rates of change are faster, increase rates by a factor of x

Stubborn: if any other waveforms are sounding, repeat the current state.

Shy: if any other waveforms are present, remain silent.

Loud-mouth: if any other waveforms are present, make the dynamic range greater than the largest of the others.

Observations on Results

1. Relatively prime cycle lengths generate richer harmonic fields than cycle lengths that are multiples of each other.
2. When the length of a state

generates a fundamental that is below 20Hz (sub-audio), the resulting sounds resemble those of heavy machinery.

3. A state can generate a temporary steady pitch if the the rates of change of its segments' sample lengths add up to zero, or close to it. We hear the repetition of the state's length (or near repetition) as a steady pitch.

4. When adjacent segments have amplitude relationships that are less than 60dB apart, their movement relative to each other seems to have no consequence on the resulting spectrum, although the lengths of the segments have a consequence on the fundamental frequency of the state. There ought to be an intelligent way to address this relationship.

5. The flexibility in sequence specification for segments, and the

possibility of segment repetition within a sequence, was done on the hypothesis that the parallel movement of separated segments could generate a second “fundamental” frequency, *i.e.*, an overtone whose amplitude was as strong as the fundamental.

This did not happen. Occasions have occurred of radical shifts in timbral presence (as I hope are hinted at by the above FFT plots), but I cannot yet successfully predict them.

6. Currently, there is no reasonable way to organize the data controlling the relationships between the waveforms. Changes in the relationships require modification of the source code. An useful organization would allow for greater flexibility of experimentation.

Acknowledgements

Herbert Brün and Keith Johnson taught me about SAWDUST, its algorithms, and its radical address to compositional premises.

Jerry Keiper and Robert Naiman, both of Wolfram Research, Inc., generously contributed their skills in mathematics and numerical analysis, and patiently answered the many questions I had on implementation.

All plots in this article were generated with *gnuplot*, version 3.0, written by Thomas Williams and Colin Kelley, and available from the Free Software Foundation. *gnuplot* ran as a child process under CounterWave, plotted the data, and converted the result

to PostScript.

The FFT algorithms are from the book *Numerical Recipes in C*.

I'd like to thank Wolfram Research, Inc., for their contributions of time and support on their computers and equipment.

I prototyped the synthesis algorithms in *Mathematica*, V2.1, then rewrote them in *C* for speed. The program currently runs on a NeXT 68040 workstation, running NeXTStep 2.1.

GRANULAR SYNTHESIS WITH INTERACTIVE COMPUTER MUSIC SYSTEM

Agostino Di Scipio
Graziano Tisato

Centro di Calcolo di Ateneo - Via S. Francesco 11, I-35121 Padova
Fax +39 49 8283733 - E-mail music01@unipad.unipd.it

Introduction

As a form of sound synthesis grounded on a microstructural representation of the musical signal, granular synthesis raises twofold set of strictly related problems, concerning 1) the inexpensiveness and effectiveness of algorithms of signal processing (oscillators, envelope generators, phase-level controls) and 2) the design of a high-level control-structure, defined as "*front-end parameter processor*" [1]. Typically, the density of grains, a determinant factor, is dependent on the inexpensiveness of the signal processing algorithms in terms of their computational load. In turn, the flexibility of controls and the degree of generality of the synthesis process are heavily dependent on the density of grains.

Of course, the higher is the density of grains and the more critical the role of high-level controls becomes. As a consequence, the characteristics of the parameter control-structure are a major factor influencing the *opacity* of granular synthesis methods.

The task of the control-structure should be conceived as the task of providing a description of *how grains overlap and repeat through*

time. Here lies the potential of granular synthesis, if considered not only as a technique of *sound synthesis*, but also as a powerful method of *microstructural time modelling of sound* [2], by which we mean a prospective of electroacoustic and computer music involving the composition of timbral events starting at the level of micro-time patterns of elementary signals. Thus conceived, granular synthesis acts as a model of materials *and* a model of musical design at once [3], a case of compositional approach and creative sonic design whose music-theoretical implications deserve special attention.

Control-structure relevance

In order to yield the superposition and juxtaposition of signals, each having its own amplitude and frequency, like

$$s(t) = \sum_{n,k} a_{n,k} g(t + kt)e^{jn\omega t}$$

(where $e^{j\omega t} = \cos\omega t + j\sin\omega t$), software synthesis may resort to different modules addressing different scales of time: signal generators, envelope generators, *local* controls pertaining to the parameters of each single grain, and more *global* controls describing the

evolution of low-level parameters throughout the synthesis process. The control-structure should interconnect all these modules. It represents, indeed, the operationalization of some theoretical model of *how* does the low-level organization and temporal displacement of myriads of grains give rise to a global, coherent acoustical behaviour [2]. This point reflects a cognitive perspective of major relevance in music: The problem of making some higher-level morphological coherence emerge from many partial details, *is* a compositional problem *tout court*, and remains a question for major importance for the composer even when it concerns the level of sound materials, i.e. when *timbre* is conceived as the central dimension of musical articulation [4]. Significantly, recent studies in the field of auditory scene analysis [5] give relief to the fact that *granular* representations can well characterize *textural* percepts and other dynamical sonic phenomena (transients, for example). One needs not only a good description of the basic grain itself (or several grains); determinant is also, if not mainly, a good description of *how* grains repeat through time [5], hence a model of how tiny irregularities coalesce in a more or less homogeneous auditory image.

Precedents of the work

After Di Scipio's long experimentation devoted to designing and testing programs for granular synthesis and processing with an IBM 80486 and the realization of 3 musical works "*ikon*"

(quad tape, 1991), "*plex*" (double-bass and quad tape, 1991) and "*Kairòs*" (soprano sax and stereo tape, 1992), finally, all software was transported and further improved by Tisato in his *Interactive Computer Music System* [6]. ICMS is a venerable (first release in 1975), powerful software system for analysis, synthesis and editing of sound, with some emphasis on linear prediction code (LPC) and methods of analysis/resynthesis of sound [7]. Today, the engine of ICMS is a IBM 9121 mainframe computer, used for time-sharing applications, which returns minutes of 4-channel audio in very few seconds. Also, one can have access to ICMS from PC and NeXT computers via Ethernet network.

Basic technical criteria

The very basic idea is that *granular synthesis* can be reduced to a particular case of *granular processing* (the **GRANULAR PROC.** options of ICMS are selected in the **SOUND PROCESSING** menu). In practice, each grain is generated following three main steps:

- 1) a pointer to a pre-existent source soundfile is calculated;
- 2) n samples are read from the file;
- 3) the sequence of n samples is processed, enveloped by a given function and written into a target soundfile. The system then jumps back to update the pointer.

a) Case of *granular synthesis*: The source file contains a signal, as a single sine wave of linearly (or exponentially) increasing frequency; the pointer, then, will be in some direct relation with the frequency of

the grain being generated. While in the linear variation case the spectral bandwidth of grains remains constant regardless of the frequency, in the exponential, it broadens as frequency is higher. The latter case better matches perceptual criteria and theoretical issues of signal representation [1], [8].

b) case of *granular processing*: If a sampled sound is stored in the source file, the pointer *history* results in a sequence of variable wave form grains, depending on the sequence of fragments read from the soundfile.

The two cases are distinct only in that completely different signals are being processed, and no other practical distinction can be drawn. This simplification radically decreases the computational load of the signal generators and allows one to experiment with different approaches to the design of the control-structure. Also, high-level strategies can be studied and tested with different source soundfiles.

Dynamical parameters controls

Clearly, the synthesized sound is dependent on the content of the source soundfile, as well as on the pointer history. The latter is itself determined by the parameter control-structure. In the ICMS granular options, the pointer moves either with linear motion (sequential access to the sourcefile) or non-linear motion (random access to the sourcefile). The pointer moves linearly in the "direction of time" only when option 1) in the GRANULAR PROC. submenu is selected (fig. 1). It is the only option, moreover, which has little sense if not used for the processing of

sampled sounds. Time-stretching and time-contracting of sound can be performed, as well as various mixtures of the two in the same process (for example, a musical gesture beginning much faster than reality and then slowing down thousands of times).

The options 2-7 refer to various methods of random access to the source file, according to the equations of fig. 1. Difference equations utilized by selecting options 4-7 represent simple iterated models with complex dynamics, i.e. featuring non-linear behaviours with both chaotic, unpredictable and more structured, recognizable patterns (plus interesting melanges of the two). The scientific literature about the topic is vast; we only refer, here, to May [9] for an introduction, and to Collet [10] for analytical insight. Similar methods have been proposed and adopted as interesting controls for granular synthesis [11], [12], [13], [14], [15], [16].

When the option 4-7 are selected, the user must declare the coefficient a of the particular difference equation, which influences the global behaviour of the iterated equation, and the initial condition x_0 , the initial state of the iteration, which determines the sequence of the following states (slightly different initial states cause completely different sequences having the same overall behaviour). In the course of the synthesis process, the coefficient a can vary if the user typed in a final value different from the starting one. The linear variation of the coefficient corresponds to moving the pointer history through different regions in the bifurcation diagram proper to the equation; different regions give

- | | |
|---|-------------------------------------|
| 1) CONSTANT-VARIABLE STEP GRANULATION | |
| 2) GRANULATION WITH BROWNIAN MOTION | $1/f^2$ noise |
| 3) GRANULATION WITH GAUSSIAN DISTRIBUTION | |
| 4) GRANULATION WITH EQ. "DISCUBIC" | $x_n = (1-a)x_{n-1} + (ax^3_{n-1})$ |
| 5) GRANULATION WITH EQ. "LOGISTIC" | $x_n = ax_{n-1}(1-x_{n-1})$ |
| 6) GRANULATION WITH EQ. "VERHULST" | $x_n = (1+a)x_{n-1} - (ax^2_{n-1})$ |
| 7) GRANULATION WITH EQ. "MAY" | $x_n = 1 - (ax^2_{n-1})$ |

Fig. 1 ICMS granular processing menu

rise to different micro-structural conditions in the generated sound. The value of the current state in the iterated model x_n is rescaled and utilized as the pointer to the source soundfile (or as a value in some synthesis parameter within the defined range). That projects in the output sequence of grains, the conditions of regularity or chaos of the iterated model, which can yield a variety of effects. Timbral events and sound textures can be synthesized according to the evolution of the difference equation adopted; one must experiment with good combinations of synthesis parameters and equation parameters to find the best appropriate mappings.

Fig. 2 shows the synthesis parameters shared by all options. For some parameters, a *tendency-mask* control is available, which makes the range of possible values change through time. Value assignment, in that case, is done using a random number generator (gaussian distribution). All this applies with parameters "grain duration", "grain delay" (time delay between two grains), "grain amplitude" (amplitude rescaling factor) and the portion of the source soundfile submitted to granulation. To avoid

unexpected effects, parameters "grain duration", "grain delay" and "scanning step" (increment of the pointer to the source soundfile) must be assigned carefully related values. The evaluation of the remaining parameters must be well "tuned" in order to minimize (or maximize) side-effects due to the periodic phase roll-off in the generated sound. A small range of random values in these parameters helps to avoid similar by-products, if necessary.

The remaining parameters are switches. If "on", they cause phase-level modifications in the generated grain, such as writing samples backwards [14], adding an amplitude offset, phase inversion and number of repetitions of the current grain before generating the next one. This last is a kind of "group delay unit" with effects similar to comb-filtering by-products; however, the amount of delay - "grain delay" - can be dynamically controlled, here, since it can be dependent on the feature particular to the selected option of granular processing.

If active, each modification is actually operated only when an internal white-noise random generator gives a number greater than 0.5 in the range 0-1; so, phase inversion, backwards writing of samples, etc. apply only for

*** GRANULAZIONE CON PUNTATORE VAR. DA EQ. "LOGISTIC"

COEFFICIENTI DI MISSAGGIO C1, C2	0.00000	1.00000
PUNTO SOVRAPPOSIZIONE FILE-LAV.	0	MSEC
CANALE FILE-AUX (1=MONO, 2=STEREO1, ECC.)	1	
LIMITI INF. FILE-AUX SU CUI GRANULARE	0	0 MSEC
LIMITI SUP. FILE-AUX SU CUI GRANULARE	5000	15000 MSEC
PERIODO MIN MODULAZIONE GRANO	10	10 MSEC
PERIODO MAX MODULAZIONE GRANO	40	70 MSEC
RITARDO MINIMO FRA I GRANI	0	5 MSEC
RITARDO MASSIMO FRA I GRANI	10	30 MSEC
RISCALAMENTO MINIMO DI AMPIEZZA	0.10000	0.50000
RISCALAMENTO MASSIMO DI AMPIEZZA	1.00000	1.70000
NUMERO DI ITERAZIONI	10000	
RETROGRADAZIONE DEL GRANO (N=0, Y=1)	1	
SOMMA AL GRANO UN OFFSET IN % AMPIEZZA	0	
NUMERO DI RIPETIZIONI DEL GRANO	1	
INVERSIONE FASE RIPETIZIONI (N=0, Y=1)	1	
ESPOLENTE DELLA FUNZIONE COSINUSOIDALE	4.00000	
PARAMETRO DI CONTROLLO EQUAZIONE (0-4)	3.74700	3.789
VALORE X0 INGRESSO EQU. (0.00001-.99999)	0.10000	

RITORNO==>"ANNUL"

HELP==>"AP1"

Fig. 2 ICMS granulation parameters with "logistic" equation

approximately 50% of generated grains. That produces aleatory micro-modulations and, together with the effects of other parameters, give the sound characteristics of "turbulence" and irregularity in the time domain. In the frequency-domain, spectral lines corresponding to the energy of grains turn to broader frequency bands and may act even as a proper *formant* structure. Perceptually richer sounds are obtained, with complex micro-time details and, perhaps, noisy components. Finally, the *p* parameter "*cosinusoidal function exponent*" determines the shape of the grain envelope as from equation

$$s(t) = 1 - \cos \omega t^p$$

Recursive processes

Each time the system writes new samples in the target soundfile, it

rescales and mixes them with already stored samples (rescaling factors are declared in the synthesis parameters submenu, see fig. 2). That gives the user the opportunity of layering no matter how many streams of grains, if rescaling and grain delay are well-studied. The superposition of entire generations of grains yields very high granular densities and gives the sound a certain smooth continuity, notwithstanding the underlying discrete-time representation. Perception detects this character of continuity, even if sound retains characteristics of roughness and richness due to irregularities in its detailed evolution. Too many superpositions, however, result in the irreversible degradation of sound, because of too many micro-modulations and a lack of local regularities in the signal.

Furthermore, interesting sounds are produced by granular

processing of previously generated streams of grains. The utilization of difference equations in a similar process, yields peculiar results when stable points arise: grains accumulate around particular moments in time, and similar behaviours in density are perceived as "spontaneous" amplitude curves, in themselves consistent but fragmented and not perfectly smooth. This possibility was explored in "*zeitwerk (l'orizzonte delle cose)*" (1992), whose realization included no amplitude envelope generators. Rather, all musical gestures were brought forth by this process of accumulation of grains and by recursive processing using larger and larger grain durations (up to 0.2 sec.). Internally articulated textures of some noisy nature were synthesized, reminding of natural, environmental phenomena. All four sections of the piece were composed using nothing but 8 sinusoidal, fixed-frequency grains (of 48, 105, 232, 511, 1124, 2473, 5442 and 11972 Hz). These basic elements were stored in a single sourcefile utilized all through the compositional process. The challenge consisted in confining sonic design to time-domain operations exclusively. No matter how different the musical events heard in the piece sound to perception, their global morphology reflects only different properties of continuity, discontinuity, linearity and non-linearity in the "local", short-term structure of grains.

A frequency-representation, to be sure, would give relief to the invariable presence of the above listed frequencies. Spectral energy, however, appears more spread

around those formant peaks then precisely concentrated. The listener cannot really grasp this stable formant structure which remains at a subliminal level for the entire duration of the piece. The four sections sound radically different, if not contrasting, in their general timbral character, exactly because of the different ways grains follow each other and overlap, i.e. because of the micro-time structural properties of the sonic materials.

A final aspect that Di Scipio should stress, concerns the sense of "*temporal horizon*" he tried to capture. Each section results from a quasi-deterministic process, and develops as long as transformations of timbre and novel gestures arise. If a "termination-point" is reached, meaning that sonic materials seem no more susceptible evolution, then the process stops and next musical section begins. For each section, this process is activated with quasi-identical initial conditions, but the resultant long-term musical structure features timbral evolutions of its own. (In this sense, the concept of *sensitive dependency on the initial conditions*, which is fundamental in chaos theory, acted as more than a suggestive metaphor in the application of the design strategy).

Notice that what he called "termination-point" was not a "goal" orienting the whole process; rather it came out of the low-level strategy only after the latter was activated. Indeed, it was impossible to foresee for how long the process would have given rise to fresh materials, i.e. to predict the temporal horizon materials.

This impossibility gives the musical a sense of continual, and not

particularly goal-oriented, exploration.

Final observations

Independently of the implementation (in the near future in a real-time version on a NeXT computer), further work should include the automation of operations that currently can be only done "by hand", including the processing of previously generated streams of grains and the layering of arbitrarily large numbers of streams. Some sort of *synthesis by rules* can be devised, based on the particular microstructural approach. In that case, a single rule may instantiate multiple operations in the realization of an entire process. This higher-level approach would represent a "step towards the abstract" for a perspective of sonic design which, by definition, is closely bounded to the level of sound materials.

References

- [1] Roads, C., *Asynchronous granular synthesis*, in "Representations of musical signals". MIT Press, 1991.
- [2] Di Scipio, A., *Microstructural time modelling of sound. A perspective of sonic design*, Proc. "2nd International Workshop on Models and Representations of musical signal", 1992.
- [3] Laske, O., *Towards an epistemology of composition*, Interface 20(3-4), 1991.
- [4] Di Scipio, A., *La musica di due culture. Tracce di una mutazione*, in "Musica e scienza. Il margine sottile", ISMEZ, 1991.
- [5] Bregman, A., *Auditory Scene Analysis*, MIT Press, 1990.
- [6] Tisato, G., *Interactive Computer Music System. Manuale operativo*. Centro di Calcolo di Ateneo, Università di Padova, 1990.
- [7] Tisato, G., *Un sistema interattivo per la sintesi dei suoni e la loro analisi mediante elaboratore*, Proc. II CIM, AIMI, 1977.
- [8] Jones, D. & Parks, T.W., *Generation and composition of grains for music synthesis*, Computer Music Journal, 12(2), 1988.
- [9] May, R., *Simple mathematical models with very complicated dynamics*, Nature (261), 1976.
- [10] Collet, P. & Eckmann, J.P., *Iterated maps on the interval as dynamical systems*, Birkhauser, 1980.
- [11] Di Scipio, A., *Composition by exploration of non-linear dynamical systems*, Proc. of the ICMC, 1990.
- [12] Di Scipio, A., *Caos deterministico, composizione e sintesi del suono*, Atti del IX CIM. AIMI/DIST, 1991.
- [13] Di Scipio, A., *An overview of digital sound synthesis by models of non-linear dynamic systems*, Bulletin of the Inst. of Mathematics and its Application, South-end-on-sea, 1992.
- [14] Truax, B., *Chaotic non-linear systems and digital synthesis. An Exploratory study*, Proc. of ICMC 1990.
- [15] Truax, B., *Real-time granular synthesis with the DMX-1000. Software documentation*. S. Fraser Univ., 1991.
- [16] Hamman, M., *Mapping complex systems using granular synthesis*, Proc. of the ICMC 1991.

BISPECTRUM OF MUSICAL SOUNDS: AN AUDITORY PERSPECTIVE

Shlomo Dubnov and Naftali Tishby

Institute for Computer Science and

Center for Neural Computation

Hebrew University, Jerusalem 91904, Israel

and

Dalia Cohen

Department of Musicology

Hebrew University, Jerusalem 91904, Israel

E-mail

dubnov@cs.huji.ac.il}

1 Introduction

The research into the realm of musical timbre has become in recent times one of the major research topics, central to the understanding of music and the musical practice itself. With more and more musicians and scientists being involved in the research, contributions considering both the artistic, psychophysical, cognitive and physical-mathematical properties of the musical signals, shed new light and understanding on this fascinating subject. In our research we chose to treat the issue mainly from the mathematical point of view, regarding the acoustic signal as a stochastic process and suggesting new aspects for its treatment. In con-

trast to other musical properties such as pitch, interval and meter, that we clearly perceive and search for their physical characterization, the timbral parameter has neither simple perceptive characterization, nor obvious physical properties. Due to these reasons, the research of timbre takes sometimes a reverse research methodology, i.e. starting with a physical-mathematical characterization of the acoustic signal we seek for its perceptual meaning. Stochastic processes, such as acoustic signals, are characterized in general by an infinite series of correlation functions. An important subset of the processes, known as Gaussian, are completely determined by their autocorrelation, or equivalently,

their power spectrum. Much of the acoustic signal processing so far is based on powerspectral properties. This is mainly because linear systems are fully determined by their effect on the spectrum, and linear systems are sufficient to describe most acoustic phenomena, and are of course easier to understand. Yet, musical instruments have highly nonlinear characteristics which affect their tone, timber, and sound quality. In this paper we suggest the use of higher-order statistics (polyspectra) [1] [2] for the analysis and evaluation of acoustic signals and instruments. In recent years bispectral methods have been applied in various signal processing fields, such as sonar, radar, image processing, adaptive filtering, etc.[3] [4]. Surprisingly, polyspectra have almost not been used so far in auditory and acoustic signal processing, primarily due to the difficulties in the estimation and analysis. These higher order correlations, known as cumulants, and their associated Fourier transforms, known as polyspectra, not only reveal all the amplitude information of the process, but also maintain the phase information. Since for Gaussian processes all the high (greater than second order) cumulants vanish, the third and fourth order polyspectra provide an indication to the non-Gaussian nature of a random process. These mathematical facts have interesting parallels in the

acoustic realizations of signals, on which we focus in this paper. Polyspectra are the natural mathematical generalization of the power spectrum and as such naturally provide the next step in acoustic research. The bispectra contain more information about the signal then the power-spectrum, particularly about it's phase. In case of skewed signals, for instance, which have non-zero bispectrum, the signal can be reconstructed from it's bispectrum up to a constant time shift, much more then can be achieved from the power spectrum alone. In addition, it is easy to analyze and manipulate the bispectral characteristics of signals in linear systems and in quadratic non-linear systems. From the physical point of view, the bispectral parameters correspond, in some models to specific mechanisms, such as characteristics of reverberant environments, or the non-Gaussian nature of a source signal passing through a resonator. This correspondence provides us with an insight to the questions of modeling particular systems, and suggests new techniques for signal manipulation and synthesis.

Finally, the acoustic perception of the bispectrum is one of the main issues of this study. We present several ideas and experiments concerning aspects of acoustic perception, sound quality, and the nature of the human auditory apparatus that are directly related to bispectra.

The following issues are discussed in this paper:

- Polyspectral criteria for "quality" design of musical instruments.
- Effects of reverberation and chorusing on the perception of tone color. Within this framework an artificial all-pass reverberator is demonstrated.
- Tone separation by means of bispectral detection-questions of timbral fusion/segregation are believed to be influenced by the presence of strong bispectral ingredient. The ear, though almost "blind" to phase, is sensitive to long term phase behavior. This phase coherence is clearly detected by bispectral estimators.

2 Mathematical Preliminaries

In this section, multiple correlations and cumulants of time signals are defined. To simplify the matters, the discussion focuses on discrete signals. Some of the important relationships and properties of finite impulse response (FIR) signals and linear, time invariant systems are briefly reviewed. The results for general non-linear systems are more complex and require the use of Volterra-Wiener system theory [6], which are beyond the scope of this presentation.

2.1 Multiple Correlations and Cumulants

Extracting information from a signal is a basic question in every branch of science. The lack of a complete knowledge of the signal exists in many physical settings due to the nature of the observed signal or the type of measurement devices. In information processing we encounter the inverse problem given the signal we want to extract information from it in order to perform basic tasks such as detection and classification. We presume that any biological information processing system acts in a similar manner. For instance, our ears perform analysis of the acoustic signal by extracting pitch and timbre information from it. To understand our motivation to study higher order correlations it is worthwhile to recapitulate briefly some of the reasons for using the ordinary double correlation. A customary assumption is that our ears perform spectral analysis of the incoming signal. Naturally, not all of the signal information is retained in our ears, and the simplest assumption is that the phase is neglected. It is well known that the amplitude of the Fourier spectrum is equivalent to the Fourier transform of the signal's autocorrelation. This double correlation in time domain is the basic type of information extracted from the signal by our ears. This information has the meaning of signal's

spectral envelope in frequency domain. Now we intend to widen the scope of acoustical analysis by suggesting the use of triple, quadratic and higher correlations, which are known also as polyspectra in the frequency domain.

The k th-order correlation, $h_k(i_1, \dots, i_{k-1})$ of a signal $\{h(i)\}_{i=0}^N$ is defined as:

$$h_k(i_1, \dots, i_{k-1}) = \sum_{i=0}^N h(i)h(i+i_1).h(i+i_{k-1}) \quad (1)$$

and frequency domain it respond to k th-order specrtum:

$$\begin{aligned} H_k(\omega_1, \dots, \omega_{k-1}) \\ = \sum_{i_1, \dots, i_{k-1}=-N}^N h_k(i_1, \dots, i_{k-1}) e^{-j\omega_1 i_1 - \dots - j\omega_{k-1} i_{k-1}} \\ = H(\omega_1).H(\omega_{k-1})H(-\omega_1 \dots - \omega_{k-1}) \end{aligned} \quad (2)$$

Under some common assumptions, the time domain correlation converges to the k th-order moment of the process. The k th-order cumulant is derived from the k th and lower order moments, and contains the same information about the process. We prefer to use cumulants in our definition of spectra since for Gaussian processes all higher then second cumulants vanish. For zero mean sequences, the second and third order moments and cumulants

coincide. Thus we arrive at an equivalent definition of the k th-order spectrum as the $(k-1)$ -D Fourier transform of the respective k th-order cumulant of the process .

Let be $y(i)$ the output of an FIR system $h(i)$, which is excited by an input $x(i)$:

$$y(i) = \sum_{j=0}^N h(j)x(i-j) \quad (3)$$

Using the definition (1) it is easy to show that :

$$\begin{aligned} y_k(i_1, \dots, i_{k-1}) \\ = \sum_{j_1, \dots, j_{k-1}=-N}^N h_k(j_1, \dots, j_{k-1})x_k(i_1-j_1, \dots, i_{k-1}-j_{k-1}) \end{aligned} \quad (4)$$

where y_k , h_k , x_k are defined as in (1). Further, employing (1) and (2) we arrive at the frequency domain relations:

$$\begin{aligned} Y_k(\omega_1, \dots, \omega_{k-1}) \\ = H_k(\omega_1, \dots, \omega_{k-1})X_k(\omega_1, \dots, \omega_{k-1}) \end{aligned} \quad (5)$$

An important property of the polyspectra is that if we are given two signals f and g that originate from stochastically independent processes and their sum signal $z = f + g$, then:

$$\begin{aligned}
 Z_k(\omega_1, \dots, \omega_{k-1}) \\
 = & F_k(\omega_1, \dots, \omega_{k-1}) + G_k(\omega_1, \dots, \omega_{k-1})
 \end{aligned} \tag{6}$$

This property is important when considering the perception of simultaneously sounding independent signals as will be discussed later.

3 Sound Quality of Musical Instruments

The first attempts to use bispectral considerations for sound quality characterization can be traced down to Gerzon [5].

For a musical tone, the power spectrum analysis shows how the fundamental frequency and its higher harmonics compose together to form the timbre of the tone. However, the power spectrum, being "phase-blind", cannot reveal the relative phases between the sound components. Although the human ear is almost deaf to the phase differences, the ear can perceive time-varying phase differences. The bispectral analyzer is the generalization of the power spectrum to the third order statistics of the signal. The bispectrum reveals both the mutual amplitude and phase relation between the frequency components ω_1, ω_2 . If sound sources are stochastically independent, their bispectra will be the sum of their separate bispectra. In order that a bispectral analyzer should be able

to recognize the characteristic signature of the sound in the bispectral plane, the excitation of a given ω_1, ω_2 should be distinguishable from the background noise. Thus, a "good" instrument is supposed to produce a maximum bispectral excitation possible for a given signal energy. Stating the problem as "can we predict the properties of a Stradivarius?", Gerzon claimed that the design requirement for a musical instrument is that "they should have a third formant frequency region containing the sum of the first two formant frequencies". Surprisingly enough this theoretical criterion seems to be satisfied by many orchestral instruments. For example, particular cases of Stradivarius violin (435 Hz, 535 Hz, 930 Hz), Contrabassoon (245 Hz, 435 Hz, 690 Hz) and English Horn (985 Hz, 2160 Hz, 3480 Hz). In a later work, Lohman and Wimitzer [11] analyzed two flutes by calculating their bispectra. Their results demonstrate that a higher intensity of the phase of the complex bispectra is achieved for the flute of good quality. This also suggests that the intelligibility of speech could be determined by looking at the bispectral signature and might be even enhanced by adding an artificial third formant to the sum of the momentary two lowest formant frequencies. Such a device can be easily constructed by means of a quadratic filter or other non-linear speech clipping system. One must note that such

a simple device will modify the spectrum also, which might be undesirable.

4 Effects of Reverberation and Chorusing

Other more subtle problems of intelligibility can be considered by looking at effects of reverberation and chorusing. Being an important musical issue, we note, quoting Erickson [9], that "there is nothing new about multiplicity and the choric effect. What is new is the radical extension of the massing idea in contemporary music, and the range of its musical applications; but a great deal more needs to be known before the choric effect is fully understood or adequately synthesized". As mentioned previously, if the sounds are stochastically independent, then their bispectra will simply be the sum of the separate bispectra. Assume a sound source with energies S_1, S_2, S_3 at frequencies $\omega_1, \omega_2, \omega_3 = \omega_1 + \omega_2$ and bispectrum level B at (ω_1, ω_2) subject to reverberation effect. Now let us assume that this effect can be modeled as a linear filter acting as a reverberator added to the direct sound. Suppose that the effect of the reverberation only is to produce a proportionate spectrum energy kS_1, kS_2, kS_3 at $\omega_1, \omega_2, \omega_3$. A plausible model for the linear filter describing the reverberator part alone could be an approxi-

mation of it's impulse response by a long sample of a random Gaussian process. According to Eq.(10), the bispectral response of such a filter is zero, which results in zero bispectrum of the output signal. The total resultant signal contains a (stochastically independent) mixture of the direct and the reverberant sound.

The spectral energy of the combined sound at $\omega_1, \omega_2, \omega_3$ will be $(1+k)S_1 (1+k)S_2 (1+k)S_3$ at $\omega_1, \omega_2, \omega_3$ and bispectrum level B at ω_1, ω_2 . Naturally the proportion of the bispectral energy to the spectral energy of the signal deteriorated. For a signal with complex spectrum $I(\omega)$ the power spectrum equals $S(\omega) = |I(\omega)|$ and the bispectrum is:

$$B(\omega_1, \omega_2) = I(\omega_1)I(\omega_2)I(-\omega_1-\omega_2).$$

Taking a bicoherence index:

$$b(\omega_1, \omega_2) = \frac{B(\omega_1, \omega_2)}{(S(\omega_1)S(\omega_2)S(-\omega_1-\omega_2))^{1/2}}$$

we arrive at a dimensionless measure of the proportionate energy between the spectrum and the bispectrum of a signal. If $b=b_{in}$ for original signal, then after reverberation $b_{out}=(1+k)^{-3/2} b_{in}$. Thus for a reverberation energy gain k, the relative bispectral level has been reduced by a factor $(1+k)^{-3/2}$ [5]. Now consider a very similar effect of chorusing. For N identical but stochastically independent sound sources the resultant spectral energies at $\omega_1, \omega_2,$

$\omega_3 = \omega_1 + \omega_2$ are NS_1, NS_2, NS_3 and the resulting bispectra is NB at ω_1, ω_2 . Comparing again the bi-coherence indexes we arrive at $b_{out} = N^{-1/2} b_{in}$ giving a relative attenuation of $N^{-1/2}$ due to this chorus effect. It is worth mentioning once again the importance of stochastic independence. The chorusing as described above might be confused with a simple multiplication of the original signal energy by a gain factor N . Such a gain is not stochastically independent and the resulting bispectrum would be augmented by $N^{3/2}$ instead of N . Only a true lack of coherence between the replicated signals will cause the resulting bispectra to be actually NB.

4.1 Experimental results

In order to demonstrate the above effects, we have performed analysis of sampled signals of solo instrument (Solo Viola) and of an orchestral section of the same instruments (Arco Violas). (The signals were recorded from a sample-player synthesizer and are believed to be true recordings of the above instruments.)

The signals have very similar spectral characteristics and the "chorusing" feature, dominantly present in the "Arco Violas" signal, cannot be extracted from the spectral information alone. It has though its manifestation in signal's bispectral contents. We plotted the amplitude of the bi-coherence index for each of the two

signals. As we can clearly see from Fig.1, there is a significant reduction of the bispectral amplitude for the "ArcoViolas" signal. Note also that the bispectral excitation pattern is different for the two signals, with the "SoloViola" signal having few clear peaks while the "ArcoViolas" has a much more spread and noisy like pattern.

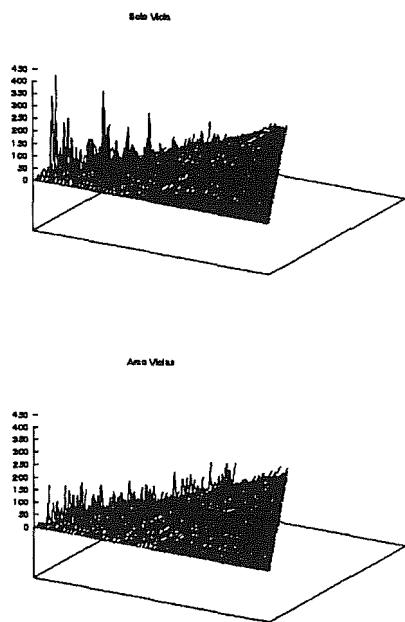


Figure 1: Bicoherence index amplitude of Solo Viola and Arco Violas signals. The x-y axes are normalized to the Nyquist frequency (8 KHz).

4.2 Artificial all-pass filter

As seen from Eq.6, the bispectra of the output signal $y(i)$ resulting from passing a signal $x(i)$ through

a linear filter $h(i)$ equals to the product of their respective bispectra. An equivalent relation holds for the linear random process, i.e. when the output signal results from passing a stationary random signal through a deterministic linear filter.

Consider now a device whose impulse response resembles a long segment of a Gaussian process. Although the filter might be on the overall deterministic, it could be considered as a random signal for any practical purpose. Applying, for instance, a bispectral analyzer of final temporal aperture to such an impulse response, would average to zero the its bispectral contents, giving us a filter with zero bispectral characteristics. Naturally, the output signal resulting from passing a deterministic signal through such a filter will have a zero bispectrum. Since the impulse response resembles a white noise signal, its spectral characteristics are flat, giving us an all-pass filter. Also, by properly scaling the impulse response we can assure that the filter gain equals 1.

The following figure describes the result of passing the original "Solo Viola" signal through a linear filter whose impulse response was created by taking a 0.5 sec. sample of a Gaussian process. The bispectral analysis of the signal was performed by averaging over 32 frames of 16 msec. each. The subjective auditory result seems to resemble a reverberation device. Fig.2 shows the

bicoherence index of the signal after filtering.

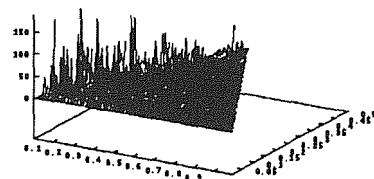


Figure 2: Bicoherence index amplitude of the output signal resulting from passing the "Solo Viola" signal through a Gaussian 0,5 sec. long filter.

5 Tone Separation and Timbral Fusion/ Segregation

Among the various questions dealing with the timbral characteristics of sounds, the problem of concurrent timbres [7] [8] is basic to the musical practise itself, manifestating itself in a daily orchestration practise, choice of instruments and the ability to perceive and discriminate individual instruments in a full orchestral sound. Originally treated in semi-empirical way by the orchestration manuals, vague criteria for evaluating orchestral choices were presented. In recent times a more quantitative acoustical studies point out several features in the temporal and spectral behavior of the sounds which are pertinent for instrument recognition and modeling spectral blend. None of these attempts have real-

ized the power of polyspectral techniques for the analysis of spectral blend. One of the most basic applications of bispectral methods reported in literature is the detection of phase coupling between harmonic components of a signal [3]. Such a phase coupling exists in most sounds produced by a single musical instrument (except idealized sine-tone generators). As claimed in section 3, the two strongest lowest frequency components of a musical signal are assumed to be harmonically related to another strong components at a higher frequency. In other words, strong coupling should exist between two harmonic components of a sound and a component at their sum frequencies. Since the power spectrum suppresses all phase relations, it cannot provide the answer. The bispectrum, however, is capable of detecting and quantifying phase coupling. An illustrative example of such an effect can be found in [3] where a 'quadratic phase coupling' phenomenon is treated, which occurs due to quadratic nonlinearities existing in a stochastic process. The non-zero bispectrum does not depend on this particular mechanism and it will hold for any case of statistical dependence between the phases. Such dependence is natural for musical sounds as mentioned above. Now having in our disposal such a powerful tool for detecting coherence between spectral components of a signal we claim that

the ear performs grouping of the various spectral components present in the sound, relating strong bispectral peaks to one source or another. Thus, spectral blend would be a blend between bispectral patterns with sounds of close bispectral signature being impossible for separation by a bispectral analyzer. Concluding this discussion we must mention that this bispectral mechanism is one among many others that influence tone color separation/blending.

6 Conclusions and Future Trends

This paper presented an important additional characteristic of the musical timbre which originates in the non-Gaussian and non-linear characteristics of the signal. The higher order spectral characteristics provide an "explanation" of various phenomena in auditory perception, which are basic to the understanding of the perception of tone color and are central to musical research and practice. Currently we are interested in applying the above ideas to a simple model that assumes a linear auto-regressive filter driven by a white non Gaussian noise (WNG) excitation. Although the details of this work are beyond the scope of this presentation, we shall mention that this model enables us to "lump" all of the non Gaussian/polyspectral properties

of the signal upon the characteristics of the input. The purpose of the work is to derive an extended feature representation of the musical signal, which would capture more signal characteristics than the standard representations such as, for instance, the widely used LPC-coefficient-based representation [12] [13] [14].

The relevance of these ideas to music lies further ahead than mathematical modeling of timbre. Direct applications lie in the field of musical tone synthesis, understanding the mechanisms responsible for timbral fusion and segregation and also in the field of musical theory. We expect that the bispectral characteristics would be closer to the musician's characterization of the 'density' or 'complexity' of the musical sound. Informally, one could say that the bispectrum contributes to some sort of 'focusing' quality of the signal, thus enabling us to distinguish between 'focused' timbre versus 'dispersed' or 'chorused' timbre. These random fluctuations between the harmonic partials of the sound add a sense of vitality to the signal, and we might suggest that there exists an analogy between this dispersion and the known meaningful variations in other musical parameters, such as the quality of intonation[15].

There is plenty of room for research in this area with several lines of investigation to be pursued. Mainly, there are many open psychoacoustical questions

that need studying and extensive experimentation. Naturally, the implications of such results to modeling of the auditory mechanism could be very substantial. Also, it is widely recognized that most of the timbral characteristics are time dependent and thus cannot be analyzed by the stationary methods discussed in this work. An extension of polyspectral methods to transient signal analysis seem to yield promising results [16]. One could study the use of nonlinear filters [17] [18] for processing of audio signals and obtaining a better control over the higher order spectra. Additionally, adaptation of the above techniques to investigation of other musical parameters is suggested. Also, it might be possible to systematize the rules of orchestration or tone-color production in orchestral and electronic music by using a bispectral description. We hope to draw a bispectral description in a manner similar to the way the spectrum-related musical staves have systematized speech. If such a far fetched ideal could be accomplished, it would provide a huge leap towards creating 'timbral composition' methodology, so desperately sought for in our days.

References

- [1] K. Haselmann, W. Munk, G. MacDonald, "*Bispectra of Ocean Waves*", in Proceedings of the Symposium on Time Series Analysis, Brown University, June 11-14, 1962 (ed. Rosenblatt), New-York, Wiley, 1963.
- [2] D.R. Brillinger, "An Introduction to Polyspectra", Ann. Math. Stat., Vol. 36, 1361-1374, 1965.
- [3] C.L. Nikias, M.R. Raghuveer, "Bispectrum Estimation: A Digital Signal Processing Framework", Proceedings of the IEEE, Vol. 75, No. 7, July 1987
- [4] J.M. Mendel, "Tutorial on Higher-Order Statistics (Spectra) in Signal Processing and System Theory", Proceedings of the IEEE, Vol. 79, No. 3, July 1991
- [5] M.A. Gerzon, "Non-Linear Models for Auditory Perception", 1975, unpublished.
- [6] M. Schetzen, "Nonlinear System Modelling Based on Wiener Theory", Proceedings of the IEEE, Vol. 69, No. 12, July 1981.
- [7] S. McAdams, A.Bregman, "Hearing Musical Streams", Computer Music Journal 3 (4) : 26-43, 60, 63, 1979.
- [8] S. McAdams, "Spectral Fusion, Spectral parsing and the Formation of Auditory Images", Ph.D. dissertation, Stanford University, CCRMA Report no. STAN-M-22, Stanford, CA., 1984.
- [9] R., Erickson, "Sound Structure in Music", Berkely, CA, University of California Press.
- [10] F.Winkel, Music, "Sound and Sensation", New-York, Dover, 1967, pp. 12 - 23, 112 - 119
- [11] A.Lohmann and B. Wirnitzer, "Triple Correlations", Proceedings of the IEEE, Vol. 72, No. 7, July 1984.
- [12] R.Cann, "A n Analysis/Synthesis Tutorial", Computer Music Journal 3(3):6-11; 3(4):9-13, 1979; and 4(1), 36-42, 1980.
- [13] P.Lansky, "Compositional Applications of Linear Predictive Coding", Current Directions in Computer Music Research, MIT Press, 1989.
- [14] R.Gray, A.H.Gray, G.Rebolledo, J.E.Shore, "Rate-Distortion Speech Coding with a Minimum Discrimination Information Distrtion Measure", IEEE Transactions on Information Theory, 27 (6), November 1981.

[15] D. Cohen, "*Patterns and Frameworks of Intonation*", Journal of Music Theory, 1969.

[16] J.R.Fonollosa, C.L.Nikias,
"Wigner Higher Order Moment Spectra: Definition, Properties, Computation and Application to Transient Signal Analysis", IEEE Transactions on Signal Processing, 41(1):245-266, January 1993.

[17] G.L.Sicuranza, "Quadratic Filters for Signal Processing", Proceedings of IEEE, 80(8):1263-1285, August 1992.

[1 8] I . P i t a s ,
A.N.Venetsanopoulos,
"Nonlinear Digital Filters",
Kluwer Academic Publishers,
1990.

REAL-TIME CONTROL OF GRANULAR SAMPLING VIA NONLINEAR PROCESSES USING THE IRCAM SIGNAL PROCESSING WORKSTATION

Cort Lippe

IRCAM, 31 rue St-Merri, Paris, 75004, France
email: lippe@ircam.fr

Introduction.

Interest in granular synthesis, along with compositional strategies for exploring this technique, have been presented by various composers [1], [2]. More recently, important compositional and technical results have been presented in the domain of real-time granular sampling [3], which has proven to be a powerful technique for timbral transformation of sampled sounds in real time. This paper discusses essential differences between granular synthesis and granular sampling techniques and describes a musical application using the IRCAM Signal Processing Workstation (ISPW) [5] in which granular sampling is controlled via nonlinear processes in a real-time compositional environment [6], [7], [8]. Compositions by the author for instruments and live ISPW are presented which make extensive use of “expressive” control of granular techniques via the detection and tracking of musical parameters of live instruments in

real time [9], [10]. Finally, a user interface, developed by the author using the program Max [11], [12] running on the ISPW, is described.

Granular Techniques.

A simple description of a granular synthesis model includes the following constants: a sinusoidal waveform, a bell-shaped amplitude envelope with a duration of 20 milliseconds, and an overlap time of 5 milliseconds between successive grains. Grains of sound, produced at a high rate of speed, are usually overlapped with neighboring grains in order to produce a certain density and continuity of sound. Pitch, maximum amplitude of individual grains, and grain density (rate of grain production and overlap of successive grains) may be considered as compositional variables in this model.

The technique of granular sampling involves the application of the above-described technique

whereby the waveform used in granular synthesis is replaced by a small chunk of sampled sound. Thus, for each grain, the onset time into a sampled sound becomes a compositional variable, along with the pitch, amplitude, and grain density. In a more detailed model of granular sound production, the waveform (or sampled sound), envelope description, grain duration, spatial location of each grain, etc., may also be considered as compositional variables.

With this large palette of available parameters, it is clear that an immense quantity of data may be required in choosing individual values for each grain of sound. Historically, compositional algorithms have often been employed to automate these choices. The practicality and necessity of automating control of granular parameters was obvious to Xenakis, who, prior to working with granular techniques in electronic music, had already explored similar problems in the instrumental domain during the 1950's with works such as *Metastaseis*, in which he employed techniques akin to a kind of "granular" conception of instrumental music.

Compositional Implications.

While granular synthesis and granular sampling are variants of the same technique, their musical essences lie at opposite poles of

the electronic music paradigm. One is immediately confronted, historically speaking, with the two main categories of electronic music: granular synthesis is *elektronische Musik*, making use of purely synthetic sounds, while granular sampling is part of the world of *musique concrète* in which recorded sounds are manipulated and transformed. As the Canadian composer, Jean Piché, has suggested, granular sampling is an "input dependent" technique. Thus, using granular techniques on sampled sounds offers an obvious level of musical implication which does not exist in granular synthesis: one is acting on and transforming a pre-existing sound object.

As mentioned above, in granular synthesis the parameters most often controlled algorithmically are the pitch, amplitude, and density of grains. While the ordering of grains in a coordinate space is calculated, giving some sort of density distribution, the concept of grains in an ordinal sense remains somewhat abstract. The results of arbitrarily different orderings yield sounding variants which, although possibly very different in nature, remain abstract synthetic sounds. Since the synthetic waveform used in granular synthesis is replaced by a small portion of a stored sampled sound in granular sampling, an additional parameter exists: onset time into the stored sound. This additional parameter can be of pri-

mary importance in granular sampling. No longer a kind of “commutative” or arbitrary parameter, *grain order* may have important consequences, creating an implicit hierarchy of parameters. Using spoken speech as a sampled sound, if onset times descend in an ordinal fashion from high to low, while density distributions of all other parameters are randomly calculated, the sounding result will always be recognized as spoken speech played backwards even though variants may sound quite different. Furthermore, one of the principle musically interesting characteristics of granular sampling is the ability to “deconstruct” sounds via the manipulation of onset times, moving between the boundaries of recognizability and non-recognizability on a continuum.

ISPW User Interface.

Using Max on the ISPW, I constructed an interface for controlling granular sampling in real time. Parameter settings can be given using sliders and number boxes, and parameters can be changed independently over time by automating control processes. Max also allows for real time switching from one sampled sound to another; either by reading elsewhere in memory, loading soundfiles from disk, or sampling anew (all of which can be done while the granular reading contin-

ues to take place). Independent granular sampling tasks can run at the same time. Each task can produce a single stream of grains or multiple, simultaneous grain attacks (the number of tasks and the number of overlapping grains within a task being limited by real-time constraints). A recent addition to the system allows for real-time mixing and sampling of the granular output of simultaneous tasks, which then may be reused as stored samples for other granular sampling tasks. This “recursive” aspect offers exponential increases in densities, and a musically “reflexive” dimension, namely, the ability to recall earlier musical material in a real-time context.

Initial Experiments.

My initial experiments with granular sampling were extremely simple: The auditory result of randomly choosing onset times into a stored sound, while producing a single stream of grains at the original pitch of the sound, is fairly statistical. Using samples of instrumental music, one has the impression with certain phrases that, for example, an entire 10-second phrase is sounding simultaneously. This is not surprising, since any onset is just as possible as any other, and, since, in using 20-millisecond grain durations with overlaps of 5 milliseconds between successive grains, more than 60 grains are produced each

second. (Increasing the overlap time between grains will greatly increase the density of grains per second.) One sampled clarinet phrase, in particular, made up of approximately 5 seconds of rapid short notes and then a 5-second held note, was noteworthy because of the omnipresence of the long note in the statistical sound mass. It was immediately obvious that the musical content of the stored sounds being operated on was not a trivial aspect of the procedure, and that mapping algorithmic calculations onto a stored sound might produce more or less successful results if the musical content of the sound was taken into account.

Nonlinear Control.

A first attempt at controlling granular sampling using nonlinear mapping was simply to choose grains statistically within defined "tendency masks" (constantly moving windows with varying sizes in which grains are statistically chosen). For instance, a window with the size of a single grain moving forward in a sound which expands to the size of a full 10 second stored sample over a specified time produces a sound which begins untransformed and, over time, becomes a statistical sound mass. These tendency masks of constantly moving window sizes and window locations can be used to read through sounds quite freely in a kind of

statistical "scrubbing" fashion, creating more or less recognizable playback of the original sounds with a rich amount of timbral variation. Random walks through the sound can be calculated and combined with control over the numerous other parameters available: pitch, amplitude, choice of stored sample, envelope description, grain duration, rate of grain production, overlap of grains, and spatial location of each grain, giving one a vast amount of transformational flexibility. My composition *Music for Clarinet and ISPW* employs granular sampling, making use of tendency masks to control virtually all parameters of granular sampling. In a work in progress, algorithms have been tried for controlling different parameters using chaotic equations. These algorithms can be predictively and easily controlled, enabling smooth transitions from the seemingly random towards stability.

Mapping Musical Expression.

The compositions mentioned above involve the use of live performers. Since the ISPW offers tools for real-time audio signal analysis of acoustic instruments for the extraction of musical parameters, another level of control over the granular sampling comes directly from the performers, giving musicians a degree of expressive control over the electronic transformations. In *Music for Clarinet and ISPW*, the sam-

pled sounds used for granular sampling are taken directly from the performed score, either sampled on-the-fly, or prerecorded and loaded into memory during performance. Continuous pitch and amplitude tracking of a performance offers musically relevant data which can be used to control aspects of an electronic score, and perceptually create coherence between the instrument and electronics. In the clarinet piece, continuous pitch data taken from the clarinet is used to control the pitch of grains, and continuous amplitude data controls the windowing of the tendency masks of certain parameters. In a work in progress, control information gathered via spectral analysis is applied to parameters, thus allowing for timbral control of the sampling by way of instrumental color changes. (See figure 1 below.)

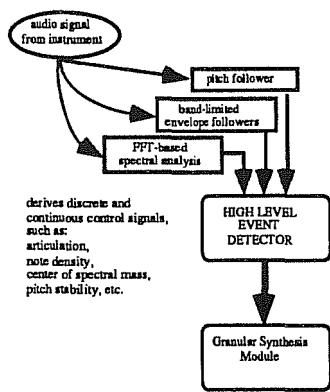


figure 1. Mapping performer expression

Conclusion.

Granular sampling is a powerful tool for transforming sampled sounds. Control of granular sampling via nonlinear processes in a real-time compositional context, and via continuous control signals made available by the detection and tracking of musical parameters of live instruments in real time, offers composers and performers a rich palette of possibilities. In addition, sampling of the output of a performer in a real time environment, while allowing the performer a certain degree of control over the granular sampling of this same material, can ultimately provide an instrumentalist with a high degree of intimate expressive control over an electronic score.

Acknowledgements.

I would like to thank Miller Puckette, Jean Piché, and Agostino Di Scipio for their invaluable technical and musical insights.

References.

- [1] I. Xenakis. *Formalized Music*. Bloomington: Indiana University Press. (Pendragon, 1991) 1971.
- [2] C. Roads. "Automated Granular Synthesis of Sound." *Computer Music Journal* 2(2):61 - 62, 1978.
- [3] B. Truax. "Real-Time Granulation of Sampled Sound

- with the DMX-1000." In J. Beauchamp, ed. *Proceedings of the 1987 International Computer Music Conference*. San Francisco: International Computer Music Association, 1987.
- [4] D. Jones and T. Parks. "Generation and Combination of Grains for Music Synthesis." *Computer Music Journal* 12(2):27 - 34, 1988.
- [5] E. Lindemann, M. Starkier, and F. Dechelle. "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features." In S. Arnold and G. Hair, eds. *Proceedings of the 1990 International Computer Music Conference*. San Francisco: International Computer Music Association, 1990.
- [6] B. Truax. "Real-Time Granular Synthesis with a Digital Signal Processor." *Computer Music Journal* 12(2):14 - 26, 1988.
- [7] R. Waschka and A. Kurepa. "Using Fractals in Timbre Construction: An Exploratory Study." *Proceedings of the 1989 International Computer Music Conference*. San Francisco: International Computer Music Association, 1989.
- [8] A. Di Scipio. "Composing with Granular Synthesis of Sound in the *Interactive Computer Music System*." In D. Smalley and N. Zahler, eds. *Proceedings of the Fourth Biennial Arts & Technology Symposium*. New London: Center for Arts & Technology at Connecticut College, 1990.
- [9] C. Lippe and M. Puckette. "Musical Performance Using the IRCAM Workstation." In B. Alphonse and B. Pennycook, eds. *Proceedings of the 1991 International Computer Music Conference*. San Francisco: International Computer Music Association, 1991.
- [10] D. Wessel, D. Bristow and Z. Settel. "Control of Phrasing and Articulation in Synthesis." *Proceedings of the 1987 International Computer Music Conference*. San Francisco: International Computer Music Association, 1987.
- [11] M. Puckette. "The Patcher." In C. Lischka and J. Fritsch, eds. *Proceedings of the 1988 International Computer Music Conference*. San Francisco: International Computer Music Association, 1988.
- [12] M. Puckette. "Combining Event and Signal Processing in the Max Graphical Programming Environment." *Computer Music Journal* 15(3):68 - 77, 1991.

AS4

A program for analysis, separation and synthesis of musical signals spectrum

Sandro Mariuz

C.S.C - DEI Università degli studi di Padova
Via Gradenigo, 6A 35100 Padova Italy
fax: +39 49 8287699

ABSTRACT

The main object for AS4 is to create a study environment for musical signals which allows to operate a great number of transformations on data coming from analysis before synthesis.

The elaboration of a sound is closely connected to its structure and, at the beginning needs a signal valuation which permits to find the main characteristics interested by the applied transformations.

An aspect sometimes not considered and which makes the difference between a synthetic sound and a real one is the presence of a noisy element in the last one (try to think to the continuous rubbing of a guitarist's fingertips when he plays his guitar or to the breath present in the flute's sound). This particular element influences, more or less, the elaborated sound quality when the used model for the representation of the original signal changes.

If we consider a real sound composed by a deterministic component and by a noisy one, with AS4 we can study the characteristics of these two components with different procedures of analysis and synthesis.

1. INTRODUZIONE

Un aspetto, spesso trascurato, che differenzia un suono sintetico da uno reale è la presenza di una componente rumorosa in quest'ultimo. Tale componente influenza infatti, in maniera più o meno sensibile, la qualità del suono elaborato al variare del modello

adottato per la rappresentazione del segnale originale. Lo studio delle caratteristiche spettrali di questa componente, unitamente alla possibilità di separarla dal resto del suono (e quindi di elaborarla in maniera indipendente) diventano aspetti essenziali per ottenere risultati qualitativamente buoni al variare della struttura del suono.

2. ANALISI E SINTESI

Dal punto di vista teorico, le due assunzioni che stanno alla base di AS4 sono:

- 1) ogni suono può essere rappresentato equivalentemente nel dominio del tempo da una forma d'onda o in quello della frequenza da un insieme di spettri;
- 2) la maggior parte dei suoni è scomponibile in una componente deterministica (che comprende la parte predicibile del suono, approssimata da una somma di sinusoidi tempo varianti) ed in una rumorosa che consiste nella parte non predicibile del suono in esame [1].

L'ambito tempo-frequenziale in cui opera AS4 consente una prima fase di analisi e sintesi basata sui modelli Short Time Fourier Transform e Sinusoidale utilizzando le tecniche di *overlapp & add* e *additiva*. Partendo da una forma d'onda si opera una finestratura, la successiva FFT consente di ottenere gli spettri del modulo e della fase su cui applicare le trasformazioni desiderate. Il tipo di finestra, la lunghezza della finestra, l'hop-size e la dimensione della FFT sono i parametri fondamentali che caratterizzano l'analisi e la sintesi. La sintesi avviene mediante IFFT seguita da *overlapp&add* (qualora si utilizzi il modello Short Time

Fourier) oppure mediante sintesi additiva (sommatoria di forme d'onda sinusoidali precedentemente individuate da un algoritmo di analisi spettrale se si usa il modello sinusoidale di rappresentazione del segnale musicale).

In fase di elaborazione (time stretching e pitch transposing), si incontrano problemi legati alla componente rumorosa presente nei suoni reali. Tale componente infatti non è ben modellabile in termini di serie di sinusoidi e produce un sensibile degrado della qualità del suono sintetizzato.

Per ovviare a tale inconveniente, considerando un suono reale composto da una componente deterministica e da una rumorosa, si è voluto operare una separazione di queste due per poter applicare le elaborazioni in modo indipendente prima della sintesi finale.

I modelli STFT e Sinusoidale, oltre ad essere essi stessi strumenti di analisi e sintesi, costituiscono la base di sviluppo per quelli Deterministico-residuo e Deterministico-stocastico che permettono questa separazione.

Se la componente rumorosa è ottenuta mediante differenza nel tempo tra segnale originale e componente deterministica (precedentemente sintetizzata mediante sintesi additiva), viene definita come *residuo*; se invece è

ottenuta con un'operazione di sottrazione in frequenza viene trattata come un *segnaletostocastico* la cui sintesi utilizza una tecnica mista basata sull'overlap&add e sulla caratteristiche teoriche di tale tipo di segnale.

Partendo da una forma d'onda si utilizza la STFT per ottenere lo spettro del segnale.

Un algoritmo di individuazione dei picchi frequentziali consente di individuare *solo* le componenti sinusoidali del suono che vengono sintetizzate mediante sintesi additiva fornendo perciò la componente deterministica.

Per ottenere il residuo, si opera una sottrazione nel tempo tra quest'ultima forma d'onda e quella del suono di partenza.

Se la separazione avviene in frequenza invece, dopo aver ottenuto la componente deterministica con il procedimento appena visto, viene applicata la STFT per ottenerne lo spettro; si sottrae infine il modulo di tale spettro a quello del segnale originale. Così facendo si ottiene il modulo dello spettro della componente rumorosa che verrà modellata con un segnale stocastico. Poiché tale segnale ne consente la completa descrizione solamente mediante le caratteristiche di ampiezza e frequenza, non è necessario

mantenere l'esatta informazione sia sulla frequenza che sulla fase istantanea.

Mediante l'individuazione dell'inviluppo spettrale (ottenuto mediante approssimazione [2] a segmenti o a gherigli) e con l'ausilio di un generatore di numeri casuali viene costruito lo spettro complesso della componente stocastica. L'applicazione della IFFT seguita da overlap&add consente poi di operarne la sintesi.

La Fig.1 mostra un esempio di separazione delle componenti deterministica e stocastica di un suono di flauto.

3. CONCLUSIONI

L'ambiente di lavoro che si viene a creare mediante l'applicazione del concetto di separazione delle due componenti del suono apre numerose possibilità di elaborazione che vanno ad aggiungersi a quelle tradizionali di alterazione delle scale dei tempi e delle frequenze. Alcuni esempi possono essere la *reverse synthesis* caratterizzata dalla presenza di una o entrambe le componenti con asse dei tempi rovesciati e la *cross synthesis* che consiste nella applicazione delle caratteristiche spettrali di un suono ad un altro per creare degli effetti di ibridizzazione. La possibilità offerta

all'operatore di intervenire sull'inviluppo spettrale della componente rumorosa, infine, consente di ottenere suoni che si differenziano da quelli degli strumenti tradizionali su cui si è operata la separazione e di conseguenza fornisce un sistema di produzione di suoni totalmente nuovi. Risultati interessanti si ottengono anche applicando il

processo di separazione alla voce. Le procedure di calcolo utilizzate in AS4 sono state sviluppate in TurboPascal e producono dei files contenenti il risultato dell'operazione effettuata. Qualora si vogliano operare delle elaborazioni con altri strumenti (phase vocoder ...) è possibile usare due procedure che consentono le conversioni numeriche desiderate.

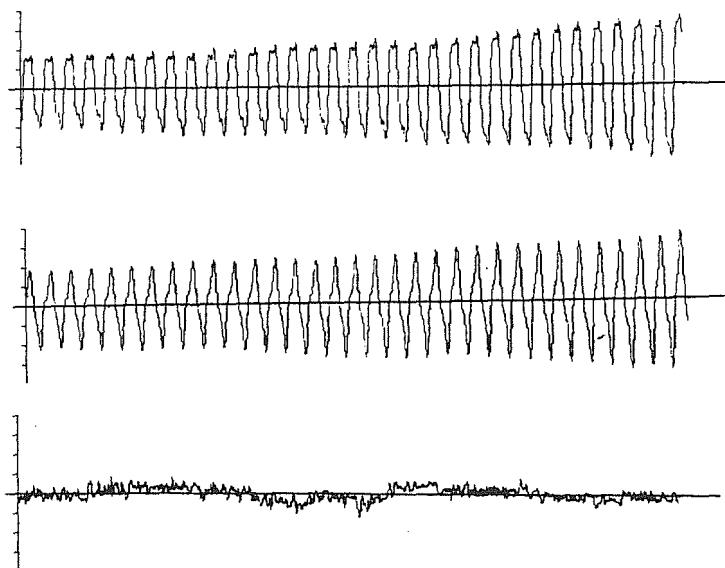


Fig.1 Suono completo, componente deterministica, componente stocastica

REFERENCES

- [1] De Poli G. and A. Piccialli and C. Roads. 1991. *Rappresentazione di musical signals*, Cambridge, Massachusetts: MIT Press.
- [2] Mariuz, Sandro. 1992. *Separazione della componente deterministica e stocastica dei suoni*, Tesi di laurea, Università di Padova, Padova.

SINTESI POLARE

Applicazioni in campo musicale di filtri digitali operanti al limite della stabilita'

Antonio Pellecchia, Alessandra de Vitis

CRM, Centro Ricerche Musicali
Via La Marmora 18, 00185 Roma

Abstract

In this paper a description is given of the fundamental aspects of scientific research as developed by the CRM - Centro Ricerche Musicali in Rome concerning sonological and musical problems.

A common requirement of either scientific than artistic field is experimentation on the audio signal, analysis and controlled manipulation of data which may occur only in the presence of powerful processes of numerical calculus.

Through informatics and technological development in the field of audio signals a substantial modification was obtained of both work methods and quality of sonological and musical research.

The technological peculiarity of the Fly30 system is described in its hardware and software characteristics as well in its possibilities of real time applications.

The central subject of the paper is the implementation of special algorithms for the processing and synthesis of acoustic signals which do not make use of pre-memorized waveforms, but rather of numerical filters operating at the limit of stability.

The paper presents the system of Polar Synthesis which is analitically described and completed by application data.

Polar Synthesis Software

In questa sezione verrà descritto il software del sistema Fly30 per l'implementazione musicale di filtri digitali operanti al limite della stabilità.

Polar Synthesis è un programma interattivo operante sotto ambiente operativo Microsoft Windows, e gestisce come hardware richiesto una interfaccia MIDI dotata di driver windows e una scheda di elaborazione numerica basata sul processore numerico in virgola mobile TMS320C30.

L'algoritmo su cui si basa il programma Polar Synthesis prende spunto dalla letteratura classica della elaborazione numerica dei segnali ed in particolare dagli studi effettuati da Klatt per la sintesi della voce attraverso filtri passa banda.

Questo algoritmo consiste nell'utilizzo di un certo numero di filtri digitali del secondo ordine posti in parallelo o in cascata. Ogni filtro digitale viene controllato in frequenza di risonanza e banda passante, tramite dei potenziometri virtuali o inserimento numerico manuale.

La sorgente eccitatrice dei filtri può essere catturata dal mondo esterno tramite microfoni, oppure può essere sintetizzata internamente. In questo caso si

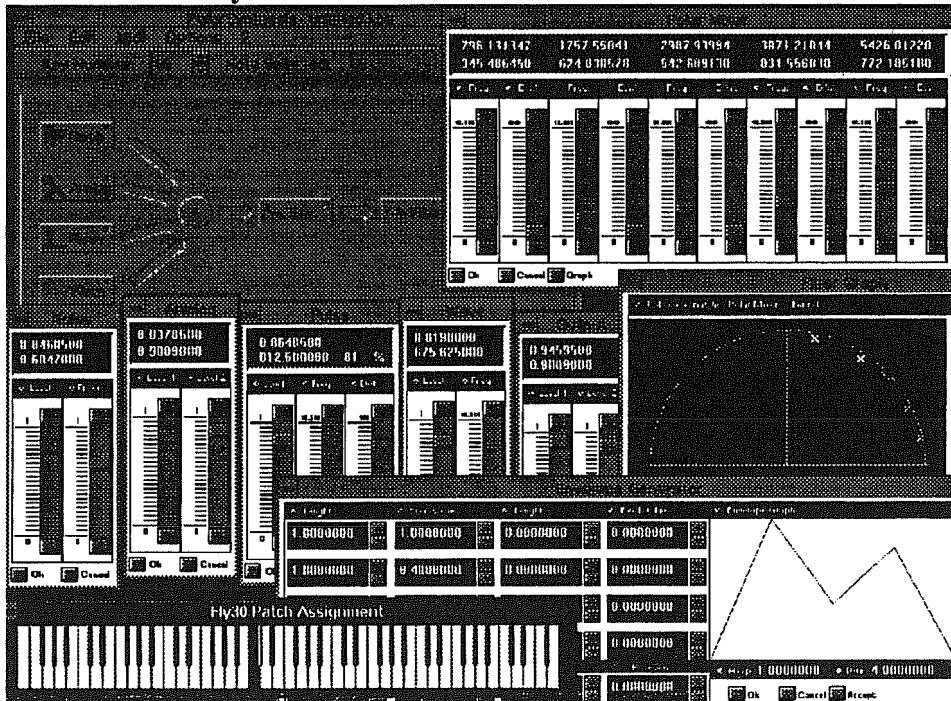
possono generare forme d'onda tramite lettura ciclica di tabelle prememorizzate o tramite un generatore di impulsi controllabile in ampiezza, frequenza e duty cycle. Un generatore di rumore viene utilizzato per dare naturalezza ed un certo grado di aleatorietà al suono generato.

Queste possibili sorgenti di suono vengono mescolate tra loro, producendo un segnale digitale che viene elaborato dal generatore di inviluppo. Questo realizza la funzione d'ampiezza nel tempo a partire dal comando di NOTEON proveniente dall'interfaccia MIDI, fino al comando di NOTEOFF relativo a quel tasto.

Questo segnale digitale viene così utilizzato per innescare le oscillazioni dei cinque filtri risonanti successivi. Ogni filtro realizza una equazione alle differenze ricorsiva del secondo ordine, composta da due ritardi e due moltiplicazioni per due coefficienti legati non linearmente alla frequenza di risonanza e alla banda passante.

Un pannello grafico visualizza il diagramma polare con le posizioni dei poli dei filtri. Tramite il mouse è possibile variare le posizioni dei poli ed ascoltare in tempo reale le variazioni di suono dovute alle variazioni dei coefficienti dei filtri.

Windows Polar Synthesis 1.0



Si puo' notare nella illustrazione in alto a sinistra la finestra principale con il disegno dell'algoritmo: diverse sorgenti sonore vengono dirette verso un generatore di inviluppo e un modulo di filtraggio per poi proseguire verso il modulo di uscita. Al centro a partire da sinistra sono visibili le diverse finestre di controllo sia numerico che tramite potenziometri virtuali dei diversi parametri dell'algoritmo: oltre ai livelli di ogni sorgente troviamo il controllo di frequenza per gli oscillatori ed il controllo del duty cycle. In basso in evidenza e'

mostrato il controllo MIDI: ad ogni tasto midi e' associato l'intero insieme di parametri associati all'algoritmo. In basso a destra troviamo il generatore di inviluppi, capace di gestire inviluppi ad 8 segmenti eventualmente ciclici, con controlli per il rilascio del tasto e per la sensibilita' alla velocita' di pressione del tasto. In alto a destra troviamo il controllo dei filtri digitali del secondo ordine: frequenza di risonanza e banda passante. I filtri digitali sono controllabili anche cambiando con il mouse la posizione dei poli corrispondenti ai filtri.

Oscillatori Polari Multipli

In questa sezione si descrivera' un metodo per sintetizzare forme d'onda canoniche quali l'onda quadra, l'onda triangolare e l'onda dente di sega tramite filtri digitali operanti al limite della stabilita' di ordine superiore al secondo.

L'oscillatore polare semplice e' in grado di generare una componente sinusoidale pura a frequenza e ampiezza dipendente dai suoi coefficienti. L'oscillazione viene generata tramite un impulso di innesto che influenzera' l'ampiezza dell'onda generata.

L'equazione alle differenze finite su cui si basa l'oscillatore polare semplice e' la seguente:

$$y_n = 2 \cos(2\pi f/fc) y_{n-1} - y_{n-2}$$

In cui fc e' la frequenza di campionamento di lavoro e f la frequenza della sinusoide generata.

Il problema consiste nel generare forme d'onda con contenuto armonico complesso senza ricorrere alla sovrapposizione di piu' oscillatori polari semplici ognuno generante le diverse componenti armoniche. Per ottenere cio' dobbiamo aumentare l'ordine della equazione alle differenze finite per poter ottenere piu' modi di oscillazione. Il numero di componenti armoniche generate sara' pari al numero di poli con parte immaginaria positiva e con

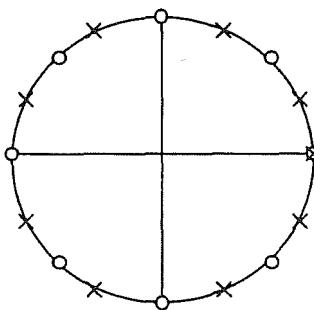
modulo unitario, e quindi avra' un valore massimo pari alla metà dell'ordine della equazione alle differenze generatrice.

Negli esempi che seguiranno si e' fatto uso di equazioni alle differenze finite il cui ordine e' stato scelto in funzione del fine esplicativo dei grafici polari. Si possono comunque realizzare equazioni alle differenze finite di ordine comunque elevato semplicemente modificando il grado delle equazioni riportate.

Onda Quadra

Sono qui riportate l'equazione alle differenze finite, la funzione di trasferimento e il diagramma polare del filtro digitale generante un'onda quadra.

$$y_n = y_{n-1} - y_{n-8} + y_{n-9} + x_n - x_{n-8}$$



$$F(z) = \frac{(1 - z^{-8})}{(1 - z^{-1} + z^{-8} - z^{-9})} = \frac{(z^8 - 1)z}{(z^8 + 1)(z - 1)}$$

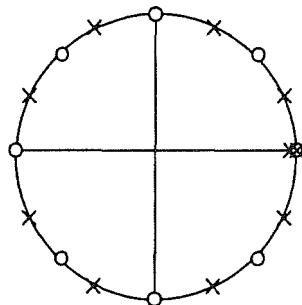
La frequenza fondamentale dell'onda generata e' proporzionale alla posizione del primo polo con

fase positiva, ed in questo caso e' pari ad un sedicesimo della frequenza di campionamento. Il polo a frequenza zero viene annullato dallo zero alla stessa frequenza e cio' garantisce l'assenza del valor medio nell'onda generata. Come si puo' notare dalla posizione dei poli, il filtro e' in grado di generare tutte le armoniche dispari della fondamentale fino alla metà della frequenza di campionamento.

Onda Triangolare

Vengono ora riportate l'equazione alle differenze finite, la funzione di trasferimento e il diagramma polare del filtro digitale generante un'onda triangolare a segno costante.

$$y_n = 2y_{n-1} - y_{n-2} - y_{n-8} + 2y_{n-9} - y_{n-10} + x_n - x_{n-8}$$



$$F(z) = \frac{(1-z^{-8})}{(1-2z^{-1}+z^{-2}+z^{-8}-2z^{-9}+z^{-10})} = \frac{(z^8-1)z^2}{(z^8+1)(z-1)^2}$$

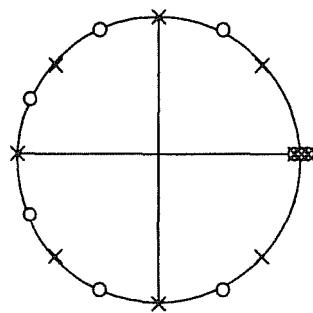
La frequenza fondamentale dell'onda generata e' proporzionale alla posizione del primo polo con

fase positiva, ed in questo caso e' pari ad un sedicesimo della frequenza di campionamento. Un polo a frequenza zero viene annullato dallo zero alla stessa frequenza, mentre il restante polo a frequenza zero determinera' il valor medio nella forma d'onda. Anche in questo caso vengono generate tutte le armoniche dispari della fondamentale fino alla metà della frequenza di campionamento.

Onda Dente di Segna

Sono qui riportate l'equazione alle differenze finite, la funzione di trasferimento e il diagramma polare del filtro digitale generante un'onda dente di segna.

$$y_n = 2y_{n-1} - y_{n-2} + y_{n-8} - 2y_{n-9} + y_{n-10} + x_n - ax_{n-1} + ax_{n-8} - x_{n-9}$$



$$F(z) = \frac{(1-az^1+az^8-z^{10})}{(1-2z^{-1}+z^{-2}-z^{-8}+2z^{-9}-z^{-10})} = \frac{(z^8-a^2+z-1)z}{(z^8-1)(z-1)^2}$$

La frequenza fondamentale dell'onda generata e' proporzionale alla posizione del primo polo con fase positiva, ed in questo caso e' pari ad un ottavo della frequenza di

campionamento. I tre poli a frequenza zero vengono annullati da tre zeri alla stessa frequenza e quindi l'onda generata e' a valor medio nullo. Come si puo' notare dalla posizione dei poli, vengono generate tutte le armoniche pari e dispari della fondamentale fino alla metà della frequenza di campionamento. Il coefficiente a deve assumere il valore $(k-1)/(k-3)$ ove k e' l'ordine del filtro (in questo caso 10).

Conclusioni

In questo articolo sono stati presentati alcuni esempi di utilizzo di filtri digitali di ordine elevato per la sintesi del suono. Questi risultati sono stati ottenuti grazie alla notevole precisione di calcolo del nostro sistema di elaborazione numerica dei segnali in tempo reale Fly30. La sintesi con oscillatori polari non presenta i problemi dovuti al troncamento e al fenomeno dell'aliasing tipici della sintesi per lettura tabellare. Inoltre l'uso di filtri digitali consente la simulazione di modelli fisici di strumenti musicali, la simulazione delle risonanze per simpatia fra elementi oscillanti come le corde, la simulazione dei transitori rumorosi negli attacchi, la simulazione degli ambienti d'ascolto e cosi' via, cose impossibili da ottenere con le sole sintesi per lettura tabellare.

BIBLIOGRAFIA

- [1] Adrien J., Caussé M. and Ducasse E., 1988, "Sound synthesis by physical models, application to strings", in Proceedings 1988 Audio Engineering Society Convention, Paris, New York.
- [2] Mathews M.V., and Pierce J.R., 1989, "Current Directions in computer music research", Cambridge: MIT PRESS.
- [3] De Vitis A., Pellecchia A., 1991 "Fly30: a DSP system for Real-time control of audio signals. Aspects of research and musical interaction", in International Workshop on Man-Machine Interaction in Live Performance, Pisa.
- [4] De Vitis A., Lupone M., Pellecchia, A., 1991 "CRM: from the Fly10 to the Fly30 system", in Atti del IX Colloquio di Informatica Musicale, Genova.
- [5] De Vitis A., Pellecchia A., 1992 "Fly30: un sistema programmabile per l'elaborazione numerica dei segnali musicali in tempo reale", in Atti del XX Convegno Nazionale di Acustica in Italia, Roma.
- [6] De Vitis A., Pellecchia A., 1992 "Music in discrete time" in International Workshop on Models and Representation of Musical Signal, Capri.

ANALYSIS , SYNTHESIS AND MODIFICATION OF PSEUDOPERIODIC SOUND SIGNALS BY MEANS OF PITCH SYNCHRONOUS TECHNIQUES

Piccialli A. , Cavaliere S. , Ortosecco I.

Dipartimento di Scienze Fisiche Universita' di Napoli
Mostra d'Oltremare PAD. 20
80125 Napoli
fax +39 81 2394508
E-mail acel@na.infn.it

Introduction

In our communication, we revisit pitch synchronous techniques for the analysis and synthesis of quasi steady-state signals, on the basis of some new results. This approach is naturally connected to the techniques of synchronous granular synthesis (see [1]), giving them both the proper tool for the analysis of natural sounds and an efficient algorithm for resynthesis and modification. It is well known that exact periodicity is never met in

real sounds both from music and voice: at most real world acoustical signals show a clear periodic basic feature; added to it we can find in pseudoharmonic sounds fast attack transients, slow amplitude and frequency modulation, and finally aperiodic components. Basic idea therefore, is to separate the underlying quasi periodic feature from the aperiodic components, what can be done by means of properly tuned comb filters or related techniques. Only after this separation both pitch detection and pitch synchronous analysis will produce proper results,

avoiding inaccuracies coming from superimposed aperiodic components. On this basis, also, the characterization of the sound under analysis and the estimate of the underlying impulse response, will carry more accurate results. For this estimate we propose a novel method of interpolation in the frequency domain, which gives good results both for resynthesis and modifications.

Finally the estimated impulse responses can be used for resynthesis and modifications, carrying both efficient and high quality reconstruction.

Separating the pseudoperiodic features from noise and transients

As far as real world pseudoperiodic signals are concerned, the underlying periodic features can be hidden by aperiodic components such as noise, transients, and even slow modulations. These components, though relevant from a psychoacoustical point of view (e.g. see [2], [3]), can introduce severe errors in the estimation of both pitch period and pitch synchronous spectral components.

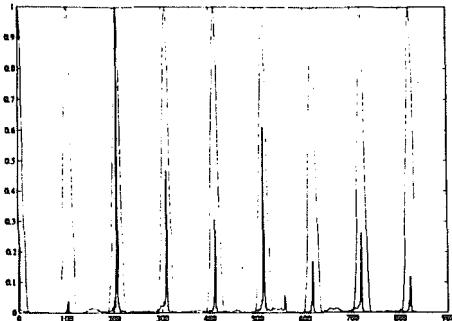


Figure 1: Typical pseudoperiodic spectrum with superimposed Comb filter

The signals to be analysed are generally time limited, and show an amplitude envelope acting as a time window on the oscillating components. The spectrum therefore is made of sharp bands tuned on a discrete set of armonically related frequencies (see fig 1).

The use of Comb filters, adaptively tuned on the frequency peaks (see fig. 1), allows the separation, on a proper scale, of the armonic signal from the aperiodic components. Psychoacoustical criteria [3], can be also used to choose the proper scale of analysis, or equivalently the proper filter bandwidth. Two different approaches can be used for filtering:

- Use of Wavelet Comb Transform [4] (multiresolution analysis), allowing the decomposition of the sig-

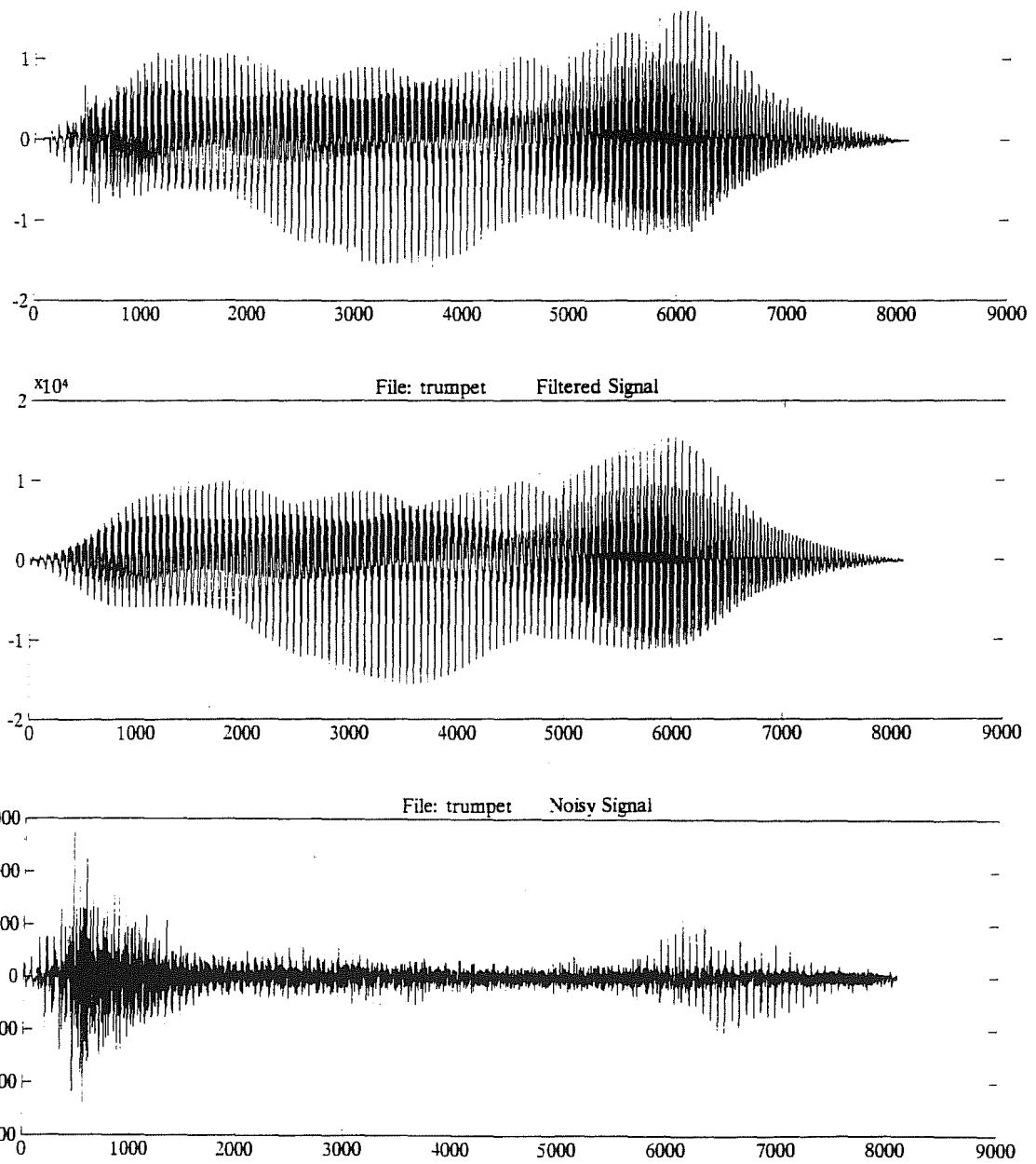


Figure 2 a

nal at different scale levels. The so called residual at the properly chosen scale is the harmonic signal, while the other part is just the aperiodic component.

- Use of classical Comb filters, tuned on the frequency peaks of the spectrum (actually on the baricenter of a frequency band); they have been designed using a prototype passband with selected features: length bandwidth and bandpass shape;

As a result from these analysis fig. 2a shows a time domain signal together with its evaluated periodic and noisy components.

Extracting pitch-synchronous grains

Second step in the analysis of pseudo armonic sounds is the extraction of pitch synchronous grains from a whole sound [5][6][7]. The above described separation eliminates errors in the evaluation of pitch periods, due to noise, fast transients and sound articulation, as stated above. Processing

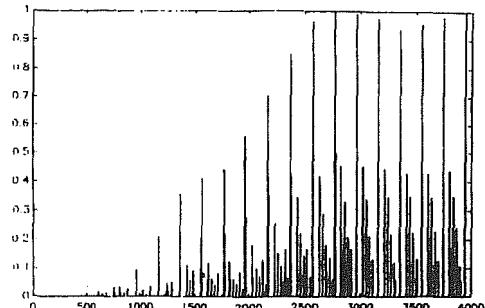


Figure 2: Energy distribution for signal [ai]

of the "clean" signal may nevertheless call for an upsample of the signal [8], useful mainly because the pitch period doesn't equal an integer number of samples. This may happen when the starting sampling rate is not high enough, and when we are analyzing high frequency signals. In this circumstance proper upsample, with interpolating filters may produce significant improvements.

After that we proceed with the determination of pitch periods. Zero crossing detection leads to the determination of local energies, computed as the absolute value of the negative to positive transitions. Referring to an underlying physical model, excitation plus filtering, energy will have local maxima where the exciting pulses are located (see fig.2). Referring to the same

figure it is a straightforward task to locate a grid in the time domain which will then ease the determination of the pitch synchronous transitions. Due to preliminary separation, this will be possible even for low energy part of the signals, or during transitions or sound/voice articulation. These transitions will allow the separation of individual frames, and will lead to a representation of the signal, made of pitch synchronous vectors.

Spectra of pitch synchronous grains and time/frequency representation

Main purpose of pitch synchronous techniques is the evaluation of the Fourier coefficients for the selected pitch period of the signal. Such analysis allows a carefull estimate of the slow evolution over time of the significant features of the signals, avoiding the use of classical analysis tools such as windowing in the time domain. Using DFT, then we can both describe the spectral content, frame after frame, or describe the evolution of each harmonic over time. The above de-

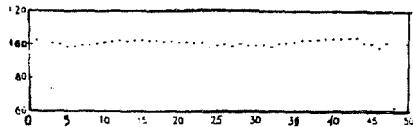


Figure 3: Pitch evolution for [ai]

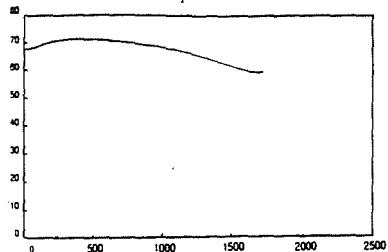


Figure 4: formant evolution for [ai]

scribed advantages due to the proposed separation are also granted here, leading to more accurate and reliable evaluation of all these parameters, discarding misleading contributions. In fig. 3 we show the evolution of the pitch and in fig. 4 the evolution of the first formant in the sound [ai].

Estimate of the pulse response of the individual grains

If the purpose of the analysis is to evaluate the parameters for resynthesis, then it is necessary to estimate the impulse response of the system under analysis (instrument or voice); that is to obtain the proper deconvolution of the sound sig-

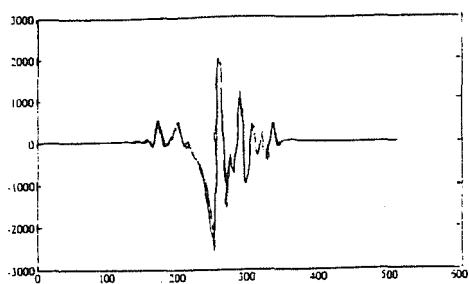


Figure 5: Impulse response for vowel [a]

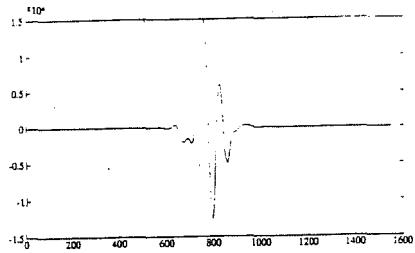


Figure 6: Impulse response for vowel [i]

nal from the exciting pulses. If the signal belongs to a class for which a source excitation model is valid, as in the case of voice, we can use well known methods from literature, such as linear prediction, homomorphic processing or Discrete All Pole modelling [9]. This is the case of fig. 5 for vowel [a] and fig.6 for [i].

In the general case of harmonic signals, we propose the use of interpolation in the frequency domain. In fact the harmonic components of singular grains, can be seen as samples of the continuous Fourier

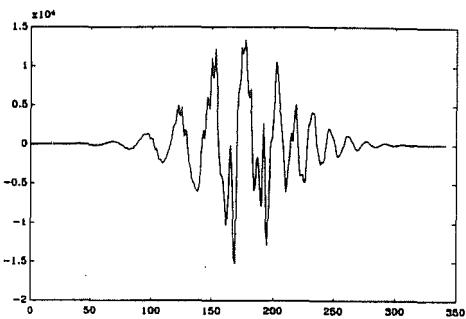


Figure 7: Impulse response for a violin tone

Transform of the system (in the case of an underlying excitation mechanism); these samples are taken at pitch intervals. Therefore, using proper interpolating laws between the known frequency samples, we can make an estimate of the complete Fourier Transform or at least of a sampled version of it, at a frequency grid finer than the spectral lines of the starting grain. Using proper interpolating laws which better embody features from the underlying physical mechanism, we can make estimates of the FT and therefore of the impulse response of the system. The interpolating law can both allow reproduction of sound and also modification of its features. We show in fig. 5 the estimate of pulse responses of violin and in fig.6 of trumpet obtained with this method.

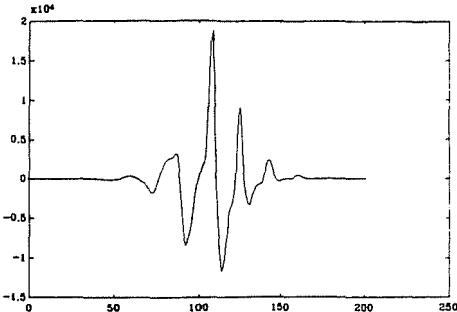


Figure 8: Impulse response for a trumpet tone

Resynthesis and modification

Resynthesis can be obtained using two channels: the first one carries pseudo harmonic signals, while the second one carries asynchronous inharmonic signals. The first part is the convolution of the impulse response with a pulse train (with proper amplitudes in order to obtain a requested sound envelope, and proper pitch evolution, to obtain a requested sound inflection and articulation). This basic mechanism requires a very low processing effort, due to the zeros in the excitation.

More complex excitation, and mul-

tipulse techniques [10] require higher computational rates but carry high quality results both for reproduction of sound and its modifications.

The harmonic part is simply superimposed with the proper amplitude envelope (it can be expressed with relatively few bits, due to the reduced dynamics, and also compacted in proper loops). Further analysis can be carried on, to obtain the features and the evolution of this aperiodic part, leading, at least in some cases, to even more efficient resynthesis .

Using the above techniques some modifications of a starting sound as time stretching or lengthening, pitch modifications and cross synthesis are easily performed with both efficiency and quality.

References

- [1] De Poli G., Piccialli A., Roads C. "Representation of musical signals" MIT Press, 1991.
- [2] Plomp R. "Aspect of Tone sensation" Academic Press, New York, 1976.
- [3] Lieberman P. "Speech physiology, speech perception and acoustic phonetics" E.Bluenstein Cambridge University Press,

- 1988.
- [4] Evangelista G. "Comb and multiplexed wavelet transforms and their applications to Signal Processing". IEEE Trans on ASSP (coming issue).
- [5] Medan Y., Eyal Yair. "Pitch synchronous spectral analysis scheme for voiced speech" IEEE Trans on ASSP, 1989.
- [6] Hess. "Pitch determination of Speech Signals" Springer Verlag, New York, 1983.
- [7] Nathan K.S., Lee Y.T., Silverman H.F. "A time-varying analysis method for rapid transitions in speech" IEEE Trans on SP Vol 39 n. 4 1991
- [8] Medan Y., Yair E. and Chazan D."Super resolution pitch determination of speech signals in speech" IEEE Trans. on SP Vol 39 n. 1 1991
- [9] Kroon P., Deprettere E.F., Sluyter R. "Regular-pulse excitation" IEEE ASSP vol ASSP-34 1986.
- [10] El-Jaroudi A., Makhoul J. "Discrete All Pole Modelling" IEEE ASSP vol 39 n 2 1991.

MULTIPLE FEEDBACK DELAY NETWORKS FOR SOUND PROCESSING

Davide Rocchesso

C.S.C. - D.E.I., Università degli Studi di Padova
via Gradenigo, 6/A 35100 PADOVA Italy
fax: +39 49 8287699 E-mail: roc@paola.dei.unipd.it

1 Introduction

Recursive comb filters are widely used in signal processing, particularly in audio applications. Digital reverberators are often based on a parallel of comb filters or on a series of all-pass filters, obtained from the comb by addition of a feedforward path. Comb structures are also used for sound synthesis in techniques derived from physical models like the Karplus-Strong algorithm [4].

In the recent past, has been underlined [2, 1] the necessity of a generalization of the comb filter through a multiple feedback delay network (MFDN) in order to obtain a sufficient time density coupled with a high frequency density in digital reverberators. I think that properties of MFDN's are largely unknown and it would be useful to know much more about these structures to evaluate the possibility of using them in a wider range of applications.

The main purpose of this paper is to explore the mathematical properties of MFDN's focusing on some possible applications, and to dis-

cuss some efficient realizations. For the moment we will focus on single-input-single-output networks, even if in many applications it is advantageous to adopt multi-input and/or multi-output; for example, this is the case of multichannel reverberators [8], where uncorrelated outputs are needed to enhance the spatial quality of sound.

2 Basic Formulation

Our MFDN is built by using N delay units. Each of them has a time length in seconds of $\tau_i = m_i T$, where $T = 1/F_s$ is the sampling period. The complete structure is reported on Figure 1 and is represented by the following relations:

$$\begin{cases} y(t) = \sum_{i=1}^N c_i s_i(t) + d x(t) \\ s_i(t + m_i) = \sum_{j=1}^N a_{i,j} s_j(t) + b_i x(t) \end{cases} \quad (1)$$

where s_i , with $1 \leq i \leq N$, are the outputs of delay lines and they are a subset of the state of this linear system. Using Z-transform, we

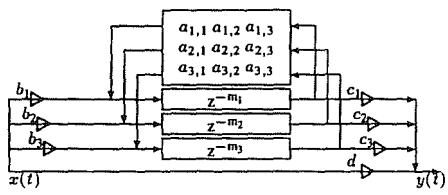


Figure 1: General Scheme for a Multiple Feedback Delay Network

can rewrite equations 1 in the following matrix representation:

$$\begin{cases} y(z) = \mathbf{c}^T \mathbf{s}(z) + dx(z) \\ \mathbf{s}(z) = \mathbf{D}(z)[\mathbf{A}\mathbf{s}(z) + \mathbf{b}x(z)] \end{cases} \quad (2)$$

where

$$\mathbf{s}^T(z) = [s_1(z) s_2(z) \dots s_N(z)],$$

$$\mathbf{b}^T = [b_1 b_2 \dots b_N],$$

$$\mathbf{c}^T = [c_1 c_2 \dots c_N],$$

$$\mathbf{D}(z) = \begin{bmatrix} z^{-m_1} & 0 & \dots & 0 \\ 0 & z^{-m_2} & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & z^{-m_N} \end{bmatrix}$$

will be called the "delay matrix" and

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{2,1} & a_{2,2} & \dots & a_{2,N} \\ \dots & & & \\ a_{N,1} & a_{N,2} & \dots & a_{N,N} \end{bmatrix}$$

will be called the "feedback matrix".

From equations 2 we can easily obtain the transfer function of our MFDN:

$$H(z) = \frac{y(z)}{x(z)} = \mathbf{c}^T [\mathbf{I} - \mathbf{D}(z)\mathbf{A}]^{-1} \mathbf{D}(z)\mathbf{b} + d. \quad (3)$$

Since $\mathbf{D}(z)$ is diagonal, $\mathbf{D}(z)^{-1} = \mathbf{D}(z^{-1})$, and we can rewrite equation 3 as follows:

$$H(z) = \mathbf{c}^T [\mathbf{D}(z^{-1}) - \mathbf{A}]^{-1} \mathbf{b} + d. \quad (4)$$

In order to obtain a state variable formulation of 1, we define

its state as the collection of the outputs of each unitary delay. Under this definition, the state-transition matrix $\bar{\mathbf{A}}$ can be written in such a way that expression 5 holds.

$$\bar{\mathbf{A}}\bar{\mathbf{A}}^T = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots \\ 0 & 1 & \dots & 0 & \dots \\ \dots & & & & \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & \mathbf{A}\mathbf{A}^T \end{bmatrix} \quad (5)$$

System poles can be found to be the solutions of either:

$$\det[\mathbf{A} - \mathbf{D}(z^{-1})] = 0 \quad (6)$$

or:

$$\det[z\mathbf{I} - \bar{\mathbf{A}}] = 0 \quad z \neq 0 \quad (7)$$

From 5 it is clear that \mathbf{A} is unitary if and only if $\bar{\mathbf{A}}$ is unitary. It follows that if \mathbf{A} is unitary, then $\bar{\mathbf{A}}$ is unitary and its eigenvalues have magnitude one; in this case, system poles are of magnitude one and the MFDN has only non-decaying eigenmodes. J.M. Jot called this particular MFDN a "reference filter".

3 Some Useful Restrictions

We can further restrict the range of possibility for matrix \mathbf{A} , imposing it to have a circulant (Toeplitz) structure. It is known that this structure allows an easy computation of its eigenvalues from Fast Fourier Transform of a row (or a column).

Consider the matrix \mathbf{A} , having a Toeplitz structure:

$$\mathbf{A} = \begin{bmatrix} a(0) & a(1) & \dots & a(N-1) \\ a(N-1) & a(0) & \dots & a(N-2) \\ \dots & & & \\ a(1) & \dots & & a(0) \end{bmatrix}$$

We can say that eigenvalues of \mathbf{A} are:

$$\begin{aligned} \{\lambda(\mathbf{A})\} &= \{\mathbf{A}(k)\} = \\ &DFT([a(0) \dots a(N-1)]^T) \end{aligned} \quad (8)$$

If we have a matrix that is unitary and circulant at the same time we can have eigenvalues on the unit circle and determine the exact phase of them. In this way we can know exactly the frequency where resonances could occur. We are not sure that each system pole corresponds to a resonance because we have not considered the effect of zeros in 4; in particular, cancellations may occur, but this will be clearer later.

Example 1

Consider a MFDN with $N = 3$ and impose the feedback matrix:

$$\mathbf{A} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{bmatrix}$$

to be unitary and circulant.

This leads to:

$$\begin{cases} a_1^2 + a_2^2 + a_3^2 = 1 \\ a_1 a_3 + a_2 a_1 + a_2 a_3 = 0 \end{cases} \quad (9)$$

There are several solutions to this system. For example we can add the condition $a_1 = a_3$, in order to reduce the complexity of multiplications. If $a_1 = a_3$ we obtain:

$$\mathbf{a}^T(z) = \frac{1}{3} [2 \ -1 \ 2] \text{ and}$$

$$\mathbf{A} = \frac{1}{3} \begin{bmatrix} 2 & -1 & 2 \\ 2 & 2 & -1 \\ -1 & 2 & 2 \end{bmatrix}$$

The eigenvalues are easily computed using a DFT:

$$\lambda = DFT \left(\left[\frac{2}{3} \ -\frac{1}{3} \ \frac{2}{3} \right]^T \right) \quad (10)$$

and they have phases:

$$\phi = \left[0 \ \frac{\pi}{3} \ -\frac{\pi}{3} \right]^T \quad (11)$$

This example will be used in some illustrations that follow. \square

Using N equal-length delay lines, where m is length, one can spread $\frac{Nm}{2}$ poles over the range $[0 \dots \frac{F_s}{2}]$ and one can compute the exact value of every pole. For the structure to be practically useful, an attenuation coefficient must be multiplied by the output of each delay line to give the feedback signal. Call

$$\mathbf{g} = [g_1 \ g_2 \ \dots \ g_N]$$

the vector of these coefficients. In the special case where all the coefficients are equal we define $g^m \stackrel{\text{def}}{=} g_i$, for every i . Unfortunately, for some choices of B and C this case reduces to a simple comb filter because of cancellations that occur in equation 4. If we change B and C , other resonances appear around the preceding peaks.

If m is constant for all delay lines, equation 6 becomes:

$$\det[\mathbf{A} - z^m \mathbf{I}] = 0 \quad (12)$$

and it is clear that system poles correspond to m -degree complex roots of $\{\lambda_i\}$. Suppose that we can arrange the system coefficients in order to have a number of resonances equal to the number of poles.

If we define the frequency density D_f as the number of resonances per Hertz, in our case we have:

$$D_f = \frac{Nm}{F_s} \quad (13)$$

Define the time density D_t as the number of pulses per second found

in the impulse response. We discriminate from the initial time density D_{ti} and the final time density D_{tf} , where the former is computed at the beginning of the impulse response, and the latter is computed after the reverberation time R_t . R_t is defined as the time involved in an attenuation of 60dB in the impulse response.

In order to obtain dense reverberation after the first reflections (i.e. after 80 msec), we could use slightly different delay lengths, for example by changing one or more sample each delay length. In this way, every mT seconds, the time density is multiplied by N.

With $N = 3$, $m = 882$ and $F_s = 44100\text{Hz}$, the frequency density is $D_f = 0.06$, usually too low for good reverberations. With $N = 8$ we could have $D_f = 0.16$, that is enough for a good reverberator.

It is possible to control the decay characteristics by modifying the transfer function of each delay unit instead of modifying the feedback matrix. In general, if one inserts a gain g_i at the output of each delay line in the MFDN, one can replace $D(z)$ with $D(z/\alpha)$ in equation 2 simply by respecting the following condition:

$$g_i = \alpha^{m_i} \quad (14)$$

The advantage of choosing nearly-equal-length delay lines is that one can control the decay time with a single coefficient.

Suppose that $g_1 = \alpha^m$ and $g_2 = \alpha^{m+x}$ where x is a variation in length. Under these assumption we have:

$$\frac{\partial g_2}{\partial x} = \ln \alpha \alpha^{m+x} \approx \ln \alpha \alpha^m$$

where the approximation is valid if

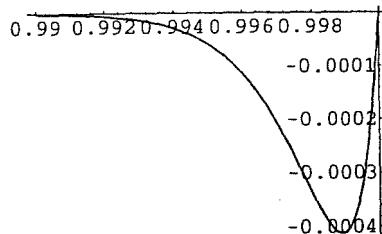


Figure 2: $\frac{\partial g_2}{\partial x}$

$$x \ll m.$$

The shape of $\frac{\partial g_2}{\partial x}$ is depicted in Figure 2, for $\alpha : 0.99 \leq \alpha \leq 1$.

The first derivative of g_2 has a minimum at $\alpha = e^{-\frac{1}{m}}$, that is equal to $-\frac{1}{m e}$ and is not lower than $4.2 \cdot 10^{-4}$.

It means that, with $m = 882$, a variation of 20 units in the delay lengths implies a maximum deviation in the exact coefficient of $8.4 \cdot 10^{-3}$. Generally speaking, if P_d is the percent of deviation on delay length, it determines a maximum of $-P_d/e$ percent of deviation in the exact coefficient.

For example, with $g^m = 0.933$, a $P_d = 1\%$ produces a deviation of decay time less than -0.107 seconds on a $R_t \approx 1.992\text{sec}$.

There is a strict relationship between MFDN's and waveguides. Consider the parallel junction of N waveguides (e.g. cylindrical tubes). Let us name $a_{j,j}$ the reflection coefficient for the j-th waveguide and $a_{k,j}$ the transmission coefficient from the j-th to the k-th waveguide. If Γ_j is the characteristic admittance of the j-th waveguide

uide, then we have [7]:

$$\begin{cases} a_{j,j} = 2\left(\sum_{i=1}^N \Gamma_i\right)^{-1}\Gamma_j - 1 \\ a_{k,j} = 2\left(\sum_{i=1}^N \Gamma_i\right)^{-1}\Gamma_j \end{cases} \quad (15)$$

In this way we obtain a scattering matrix that is symmetrical. If we properly scramble the order of incoming and outgoing waves at the parallel junction of three equal impedance waveguides, the matrix A of Example 1 can be viewed now as a scattering matrix.

Some better feedback matrices may be found, departing from a physical point of view and looking at the frequency response of the filter. In order to obtain colorless reverberation we would like to avoid clustering of resonance peaks. In principle, this could be achieved by choosing equally spaced eigenvalues for the feedback matrix, but this choice would lead to an identity matrix, that simply corresponds to a parallel of comb filters. This solution must be neglected because of its poor time response.

The latter observation looks like another form of the well-known "Indeterminacy Principle", always found in time-frequency analysis. In order to achieve a dense time response, we must look at solutions where the elements of the feedback matrix are all significantly non zero.

Example 2

A good solution for $N = 3$ is the following:

$$\mathbf{a}^T(z) = \frac{1}{3} [\alpha \ -1+\alpha \ 1+\alpha] \quad \text{and}$$

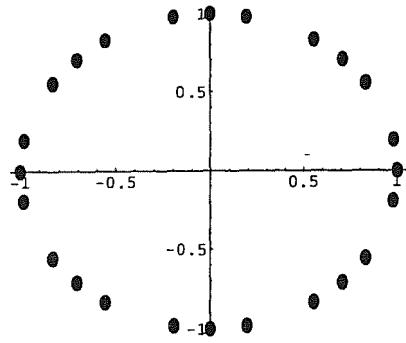


Figure 3: Distribution of Poles for Example 2

$$A = \frac{1}{3} \begin{bmatrix} \alpha & -1+\alpha & 1+\alpha \\ 1+\alpha & \alpha & -1+\alpha \\ -1+\alpha & 1+\alpha & \alpha \end{bmatrix}$$

where $\alpha = \frac{1}{\sqrt{3}}$. This choice gives the distribution of poles of Figure 3, given a delay length $m = 8$. \square

4 Hints for Implementation

In section 3 we have shown a general way of finding feedback matrices, regardless of their dimension. The circulant structure offers the possibility to find eigenvalues easily, and is suitable to a VLSI implementation because it involves a repetitive pattern of operations. Nevertheless, even under this restriction, the product of a $N \times N$ matrix by a N -dimensional column involves N^2 multiplications/additions.

For low values of N it is possible to find matrices allowing a number of operations to be linear in N . This

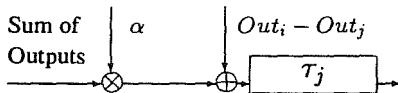


Figure 4: Operations Associated with Every Delay Line of Example 2

is the case of the matrices of Examples 1 and 2. For the matrix of Example 2 we can arrange the operations associated with each delay line as in Figure 4.

J.M. Jot [3] proposes to use matrices that are a generalization of that of example 1 to any given dimension. We have just underlined the fact that they are related to the parallel junction of N waveguides. This choice suffers from the fact that for increasing dimensions the matrix tends to become an identity matrix, thus providing an unsatisfying temporal behavior. To overcome this drawback Jot proposes the use of block unitary matrices. Our approach of studying the position of eigenvalues via a short FFT allows one to experiment with many different structures for an arbitrary dimension, in order to find both an efficient implementation and a good distribution of poles. Sometimes it could be convenient to slightly depart from reference filters by choosing matrices whose eigenvalues are not exactly one in absolute value. This choice can be justified by fairly simple involved operation and it does not give a dramatic change in spectral evolution, at least in the case of slight perturbations of a reference filter. Furthermore, starting from eigen-

values that are strictly inside the unit circle it is possible to design a MFDN without attenuation coefficients, thus saving N multiplications. In this case we can control reverberation time by using a table look-up that gives values for N coefficients, for each discrete value of the control variable.

A simple way of simulating frequency dependent losses is the use of interpolated delay lines. Some digital sound processors implement such delays in an efficient manner [6]. By choosing the interpolation coefficient one obtains a different amount of dissipation over frequency, having the maximum filtering effect at half the distance between two samples. This technique preserves the linearity of phase by giving a fractional constant delay at all frequencies. It also helps to increase temporal density by reducing the superimposition of echoes.

5 Simulating Resonators

MFDN's with short delay lines may be used to produce resonances irregularly spread over frequency. A possible application could be the simulation of resonances in the body of a string instrument.

A well-known work by Mathews and Kohut [5] has shown that in this kind of simulation, the exact position and height of resonances is not important; on the contrary they stated that:

- the peak frequencies must be irregularly spaced with respect to the harmonic frequencies of any tone.
- the Q's of the resonances must be sufficiently large so the response curve is "steep" almost ev-

everywhere, that is to say, the magnitude of the derivative of the impulse response curve must be large almost everywhere.

- the peaks must be sufficiently close together so the depth of the intervening valleys does not exceed about 15dB.

In Figure 5 are reported the frequency responses of the filter having the matrix A of Example 1 and $d = 0$, for different values of delay lengths, B and C .

Acting on delay lengths it is possible to move poles in frequency, while acting on B and C we can re-shape the frequency response. The gain g determines the maximum peak to valley distance.

Using unitary and circulant matrices we can compute the exact position of system poles under a small delay length deviation.

System poles are the solutions of equation 6 that can be written in terms of algebraic complements $D_{i,j}$:

$$\begin{aligned} \det[A - D(z^{-1})] &= \\ (a_{1,1} - z^{m_1})D_{1,1} - a_{1,2}D_{1,2} + \dots &= 0 \end{aligned} \quad (16)$$

Put $m_1 + \delta m$ in place of m_1 and obtain:

$$\begin{aligned} (a_{1,1} - z^{m_1})D_{1,1} - a_{1,2}D_{1,2} + \dots \\ + z^{m_1}(1 - z^{\delta m})D_{1,1} = 0 \end{aligned}$$

If λ is a system pole using m_1 , suppose $\lambda + \delta\lambda$ a pole using $m_1 + \delta m$. With the approximation:

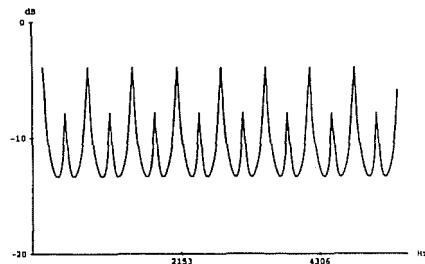
$$(\lambda + \delta\lambda)^{m_1} \approx \lambda^{m_1} + m_1\lambda^{m_1-1}\delta\lambda \quad (17)$$

the equation 16 becomes:

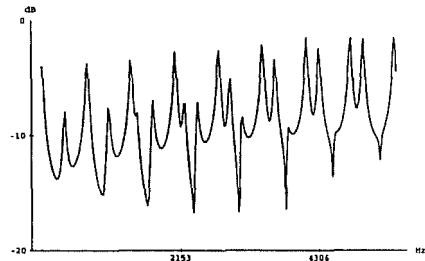
$$m_1\delta m\lambda^{-1}\delta\lambda^2 + (m_1 + \delta m)\delta\lambda - \lambda^{1-\delta m} + \lambda \approx 0 \quad (18)$$

And this equation can be solved to give an approximated formula for $\delta\lambda$:

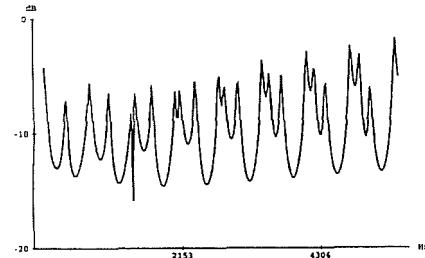
$$\delta\lambda \approx \frac{m_1 + \delta m \mp \sqrt{\Delta}}{2m_1\delta m\lambda^{-1}} \quad (19)$$



Delays [Samples]=[64 64 64];
B=[0.7 0.6 -0.7]; C=[0.8 -0.1 0.5]



Delays [Samples]=[64 59 67];
B=[0.7 0.6 -0.7]; C=[0.8 -0.1 0.5]



Delays [Samples]=[64 62 67];
B=[0.1 1.0 -0.7]; C=[0.8 -0.1 0.5]

Figure 5: Frequency Responses for Different Values of B and C

where Δ is the discriminant of equation 18 and the choice of $-$ or $+$ sign must be done accordingly to the fact that a delay expansion implies a frequency contraction ($\delta f < 0$), while a delay contraction implies a frequency expansion ($\delta f > 0$).

In a practical case, we can give a worst-case value to the deviation $\delta\lambda$ considering a deviation δm equal to the maximum deviation for each delay line.
In this case we have a change in every peak frequency equal to:

$$\delta\lambda = \left(\frac{1}{m+\delta m} - \frac{1}{m} \right) m\lambda$$

References

- [1] M.A. Gerzon "*Unitary (Energy Preserving) Multichannel Networks with Feedbacks*" Electronics Letters V 12(11) 278-279 1976.
- [2] J. M. Jot. and A. Chaigne "*Digital Delay Networks for Designing Artificial Reverberators.*" 90th Convention of the Audio Engineering Society. Paris, Feb. 1991.
- [3] J. M. Jot. "*Etude et Réalisation d'un spatialisateur de son*" PhD Thesis. TELECOM Paris 92 E 019. Paris, Sept. 1992.
- [4] K. Karplus and A. Strong "*Digital Synthesis of Plucked-String and Drum Timbres*" Computer Music Journal 7(2): 43-55 1983.
- [5] M. Mathews and J. Kohut "*Electronic Simulation of Violin Resonances*" The Journal of the Acoustical Society of America 53(6): 1620-1626 1973.
- [6] A. Paladin and D. Rocchesso "*A Dispersive Resonator in Real-Time on MARS Workstation*" International Computer Music Conference. San Jose - California, Oct. 1992.
- [7] J.O. Smith "*Music Applications of Digital Waveguides*" CCRMA Report No. STAN-M-39 1987
- [8] J. Stautner and M. Puckette "*Designing Multichannel Reverberators*" Computer Music Journal 6(1): 52-65 1982.

A REAL TIME CLARINET MODEL ON MARS WORKSTATION

Davide Rocchesso, Fabio Turra

C.S.C. - D.E.I., Università degli Studi di Padova
via Gradenigo 6/A 35100 PADOVA, Italy
fax +39.49.8287699 E-mail roc@paola.dei.unipd.it

This paper presents the development of a workbench for studying a model of clarinet. The purpose was to verify the existing models and to study their possible simplifications, and to obtain a real time realization, suitable for an interactive use. Moreover, starting from clarinet model, we wished to obtain general computational schemes for wide instruments families, without leaving physical models representation.

The model was implemented on MARS (IRIS, Rome), a workstation based on a board containing two DSP X20, digital audio signal processors, operating in fixed point. The whole work was done using the assembly language of the processors, in order to obtain a high level of parallelism.

The starting model is a clarinet model widely tested in literature [1]. The clarinet is divided into two blocks: an exciter, the reed, and a resonator, the bore. The bore is divided into a cylindrical tube and a flare, the bell. The variables interacting between exciter and resonator are the incoming and the outgoing pressure waves, travelling inside tube. The tube behaves as a lossless waveguide, and can be simulated by a simple delay line.

The bell behaves as a couple of filters, reflecting low frequencies inside tube, with a sign change, and transmitting high frequencies outside.

In an early realization we adopted a static reed model, where the air flow depends only on the pressure difference on the reed, and can be obtained by a table lookup. The input variables are incoming pressure and mouth pressure. The output variable is the outgoing pressure, obtained by addition of the incoming pressure and the product of the air flow and the acoustic impedance of the tube.

The base model was gradually enriched. The first improvement consisted in using a dynamic model of the reed, i.e. with memory [2]. In this model a differential equation represents reed dynamics. The reed position at every time sample is obtained solving this equation by means of back finite differences. A special block controls the non-linearity introduced by the closed reed situation: when the reed beats on the mouthpiece, it nullifies reed velocity and reed acceleration (we hypothesize anelastic bump). The information relative to reed velocity, in this quantized model, is brought by the reed position at the previous sample instant.

Air flow calculation is based on an experimental law for wind instruments, which gives a relation between the reed position x , the pressure gap on the reed dp and the air flow u through the reed narrow. In the case of the clarinet that law is:

$$u = A \cdot |dp|^{2/3} \cdot x^{3/4} \cdot \text{sign}(dp),$$

where A is a physical constant typical of woodwind instruments. In this equation air inertance should be added, but its effect is the same as a lowpass filter, and it can be included in the effect of resonator filters. The air flow calculated using the former equation must be added to the air flow pushed by the reed in its movement. All these relations are summed up in a non linear equation, which can be solved by numeric methods. The values of air flow relative to given values of pressure gap, reed position and reed velocity are written in a three-dimensional look-up table. Figure 1 shows a view of air flow variation (Zu), depending on pressure gap (dp) and reed

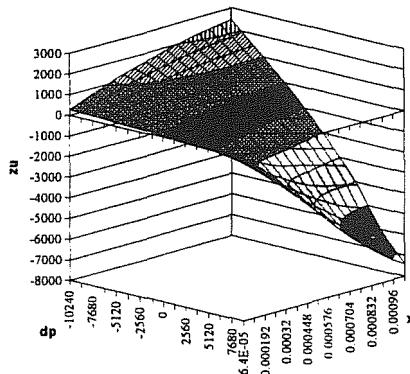


Figure 1. View of air-flow variation

displacement (x), for a given constant value of previous step reed displacement (x_1).

The three-dimensional look-up table was implemented on DSP using three tables. The minimum dimension needed, for having no advisable loss of sound quality, is of 8 kwords.

A second improvement of the model, which brought a further increase of realism, was the introduction of a noise generator. In fact, the sounds produced by actual instruments are not exactly periodic, but they contain a noise component. That noise is a minimum part of sound, but it is a very significant part, and its absence can give a sensation of artificiality. In a woodwind instrument, the noise is caused by turbulent air motion, producing random pressure variations, depending on flow velocity and reed displacement. A noise generator model, developed for human voice [3], was adapted to clarinet, applying it to the narrow produced by the reed. In that model remarkable simplifications may be introduced, neglecting the dependence on reed position and approximating the internal resistance of the noise generator as a constant, so that it can be included in tube impedance. A comparison between original model output and approximated model output showed the validity of simplifications. The noise pressure can be represented, after all, as a pseudo-random sequence having a non linear function of air flow as its envelope. Air flow is filtered by a lowpass filter, to ensure stability. In figure 2 a block scheme of the model of

dynamic reed with noise generator is shown. The variables shown in the picture are the mouth blowing

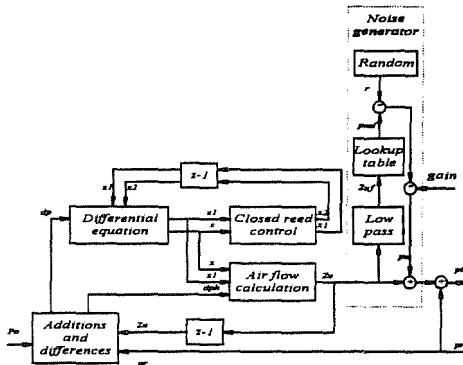


Figure 2. Model of dynamic reed

pressure (P_o), the pressure gap on the reed (dp), the reed position at present time (x) and one or two samples before (x_1, x_2), the noise pressure (p_n), the incoming pressure wave (p_i), the pressure wave reflected by the tube (p_r) and the air flow (Z_u) normalized to pressure dimensions.

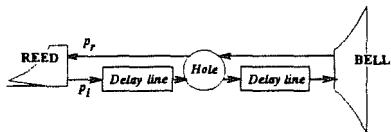


Figure 3. Scheme of the instrument

At last, the clarinet model was enriched by adding a hole, that can be gradually closed and moved along the tube. For the hole, we adopted a model which takes into account the effects of a key (or a finger) positioned above hole aperture [4]. The hole behaves as a discontinuity of the tube and produces, on a travelling

pressure wave, a reflected wave and two transmitted waves, into the tube and into the hole. This kind of junction is simulated adopting a lattice scheme [5] and introducing a lowpass filter to realize the frequency dependent reflection coefficient ρ . As ρ depends also on distance between the key and the hole, filter band is made variable: the gradual hole closure corresponds to the gradual closure of the filter. The reflection coefficient is negative, so the scheme contains also a sign change.

The hole can be moved along the tube by varying the length of both the delay lines simulating the tube, as illustrated in figure 3.

The scheme here individualized presents a good degree of generality, because it can be easily suited to the simulation of other wind instruments (oboe, trumpet, etc.) and, with some effort but without structural modifications, also of instruments of other families (strings, etc.). The choice of the table lookup for the solution of non linear equations, beyond making calculation more efficient, facilitates instrument changes, requiring only tables substitution [6]. The non linear relation seen for the clarinet becomes valid for double reed woodwinds by just changing the exponents of pressure gap and reed position. By also changing the sign of the constant form of the differential equation, it is also possible to simulate brass instruments [2]. The most interesting point is that it is possible, modifying the non linear equation (i.e. changing the three-dimensional look-up table),

to obtain new woodwinds which have intermediate characteristics between single reed and double reed instruments.

The insertion of a hole in the resonator is able to produce some subtle phenomena that can be encountered in real instruments, particularly during transition phases, and that cannot be obtained with simpler models.

It is interesting the result given by this model when simulating the gradual opening of the hole. Figure 4 shows the evolution of the spectrum and the waveform of the pressure wave travelling inside the tube, during hole opening. The hole was, in this case, placed quite near the reed, to exacerbate the frequency change (almost to the fifth harmonica).

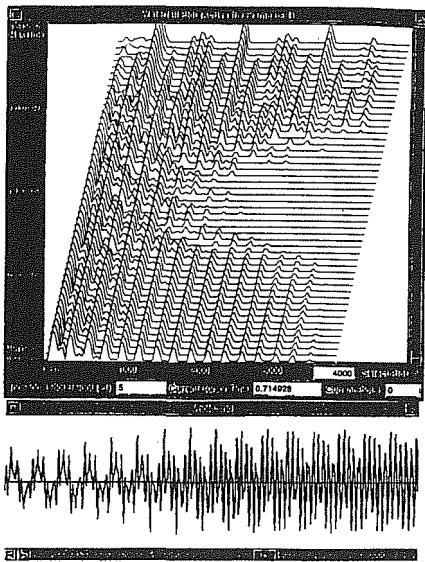


Figure 4. Evolution of the spectrum and the waveform during hole opening

The final result is a satisfying model of clarinet, where different

parameters were made available to experiment real time variations, in order to make a gradual and intuitive change of instrument timbre possible.

References.

- [1] M. E. Mc Intyre, R. T. Schumacher, J. Woodhouse: "*On the oscillations of musical instruments*", Journal of the Acoustical Society of America 74(5): 1325-1345, 1983.
- [2] D. H. Keefe, M. Park: "*Tutorial on Physical Models of Wind Instruments: II, Time Domain Simulations*", Seattle: University of Washington, Systematic Musicology Technical Report No. 9002, 1990.
- [3] M. M. Sondhi, J. Schroeter: "*A hybrid time-frequency domain articulatory speech synthesizer*", IEEE Transactions on Acoustic Speech and Signal Processing, 35(7): 955-967, 1987.
- [4] G. R. Plitnik, W. J. Strong: "*Numerical method for calculating input impedances of the oboe*", Journal of the Acoustical Society of America, 65(3): 816-825, 1979.
- [5] G. Borin, G. De Poli, S. Puppin, A. Sarti: "*Generalizing physical models timbre class*", Proc. Int. Conf. on Physical Models, Grenoble, 1990.
- [6] D. Rocchesso, F. Turra: "*A generalized excitation for real-time sound synthesis by physical models*", Stockholm Music Acoustics Conference, Stockholm, 1993.

FFT-BASED RESYNTHESIS FOR REAL-TIME TRANSFORMATION OF TIMBRE

Zack Settel & Cort Lippe

IRCAM, 31 rue St-Merri, Paris, 75004, France
email: settel@ircam.fr & lippe@ircam.fr

Introduction.

The Fast Fourier Transform (FFT) is a powerful general-purpose algorithm widely used in signal analysis. FFTs are useful when the spectral information of a signal is needed, such as in pitch tracking or vocoding algorithms. The FFT can be combined with the Inverse Fast Fourier Transform (IFFT) in order to resynthesize signals based on its analyses. This application of the FFT/IFFT is of particular interest in electro-acoustic music because it allows for a high degree of control of a given signal's spectral information (an important aspect of timbre) allowing for flexible, and efficient implementation of signal processing algorithms.

This paper presents real-time musical applications using the IRCAM Signal Processing Workstation (ISPW) [1] which make use of FFT/IFFT-based resynthesis for timbral transformation in a compositional context. Taking a pragmatic approach, the authors have developed a user interface in the Max programming environment [2]

for the prototyping and development of signal processing applications intended for use by musicians. Development in the Max programming environment [3] tends to be simple and quite rapid: digital signal processing (DSP) programming in Max requires no compilation; control and DSP objects run on the same processor, and the DSP library provides a wide range of unit generators, including the FFT and IFFT modules. Techniques for filtering, cross-synthesis, noise reduction, and dynamic spectral shaping have been explored, as well as control structures derived from real-time signal analyses via pitch-tracking and envelope following. These real-time musical applications offer composers an intuitive approach to timbral transformation in electro-acoustic music, and new possibilities in the domain of live signal processing that promise to be of general interest to musicians.

The FFT in Real Time.

Traditionally the FFT/IFFT has been widely used outside of real-

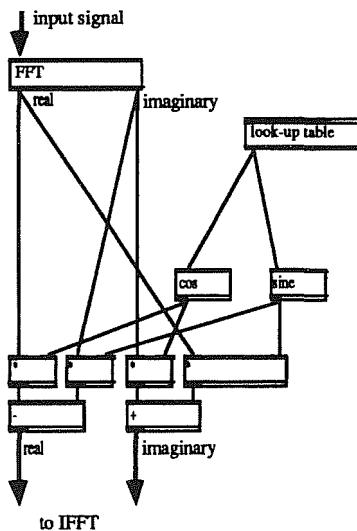
time for various signal analysis/re-synthesis applications that modify the durations and spectra of pre-recorded sound [4]. With the ability to use the FFT/IFFT in real-time, live signal-processing in the spectral domain becomes possible, offering attractive alternatives to standard time-domain signal processing techniques. Some of these alternatives offer a great deal of power, run-time economy, and flexibility, as compared with standard time-domain techniques [5]. In addition, the FFT offers both a high degree of precision in the spectral domain, and straightforward means for exploitation of this information. Finally, since real-time use of the FFT has been prohibitive for musicians in the past due to computational limitations of computer music systems, this research offers some relatively new possibilities in the domain of real time.

Algorithms and Operations.

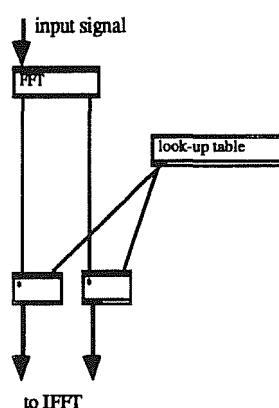
All of the signal processing applications discussed in this paper modify incoming signals and are based on the same general DSP configuration. Using an overlap-add technique, the DSP configuration includes the following

steps: (1), windowing of the input signals, (2) transformation of the input signals into the spectral domain using the FFT, (3) operations on the signals' spectra, (4) resynthesis of the modified spectra using the IFFT, (5) and windowing of the output signal. Operations in the spectral domain include applying functions (often stored in tables), convolution (complex multiplication), addition, and taking the square root (used in obtaining an amplitude spectrum); data in this domain are in the form of rectangular coordinates. Due to the inherent delay introduced by the FFT/IFFT process, we use 512 point FFTs for live signal processing when responsiveness is important. Differences in the choice of spectral domain operations, kinds of input signals used, and signal routing determine the nature of a given application: small changes to the topology of the DSP configuration can result in significant changes to its functionality. Thus, we are able to reuse much of our code in diverse applications. For example, though functionally dissimilar, the following two applications for phase rotation and filtering differ only slightly in terms of their implementation. *see following figure:*

phase rotation



filtering

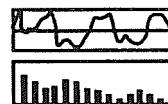


Applications.

High-resolution filtering

Highly detailed time varying spectral envelopes can be produced and controlled by relatively simple means. A look-up table can be used to describe a spectral envelope in the implementation of a graphic EQ of up to 512 bands. The spectrum of the input signal is convolved, point by point, with the data in the look-up table, producing a filtered signal. Using a noise source as the input signal, it is also possible to do *subtractive synthesis* efficiently. Because we are able to alter the spectral envelope in real time at the control rate (up to 1kHz), we may modify our spectral envelope graphically or algorithmically, hearing the results immediately. *see following figure:*

signal A



spectral envelope (signal B)

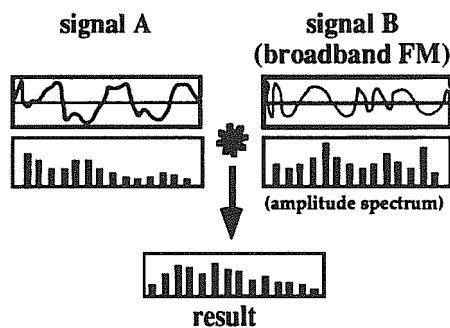


(convolution)
↓
result

Low dimensional control of complex spectral envelopes

The spectral envelope used in the above filtering application can also be obtained through signal analysis, in which case a second input signal, signal B, is needed. Signal B is analyzed for its spectral envelope, or amplitude spectrum, that describes how signal A will be filtered. Obtaining a spectral envelope from an audio signal offers several interesting possibilities:

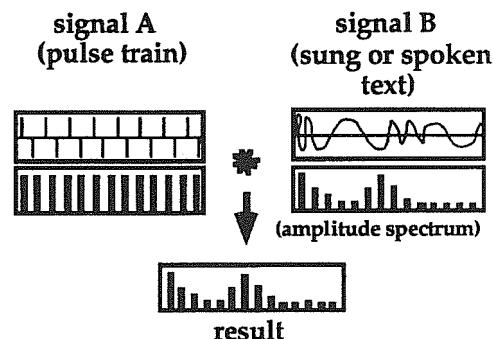
spectral envelopes can be "collected" instead of being specified, and can change at a very fast rate (audio rate), offering a powerful method of dynamic filtering. Also, audio signals produced by standard signal processing modules such as a frequency modulation (FM) pair (one oscillator modulating the frequency of another) are of particular interest because they can produce rich, easily modified, smoothly and nonlinearly varying spectra [6] which can yield complex time varying spectral envelopes. In place of FM, other standard signal processing modules can be used that offer rich varying spectral information using relatively simple means with few control parameters. One of the advantages of using standard modules is that electronic musicians are familiar with them, and have a certain degree of control and understanding of their spectra. *see following figure:*



Cross synthesis

In this application two input signals are required: signal A's

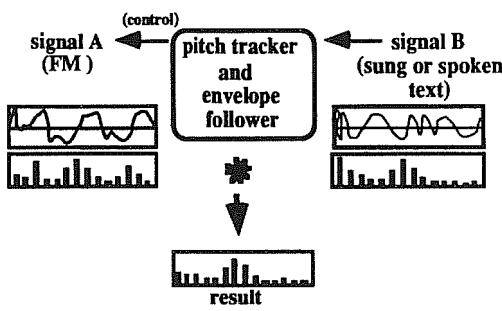
spectrum is convolved with the amplitude spectrum of signal B. Thus, the pitch/phase information of signal A and the time varying spectral envelope of signal B are combined to form the output signal. Favorable results are produced when Signal A has a relatively constant energy level and broadband spectrum, and when signal B has a well defined time varying spectral envelope. In the following example of a *vocoder*, text can be decoupled from the speaker or singer's "voice quality", allowing one to modify attributes of the voice such as noise content, inharmonicity, and inflection, independently of the text material. *see following figure:*



Mapping qualities of one signal to another

A simple FM pair may be used to provide an easily controlled, constant-energy broadband spectrum for use in cross synthesis as signal A. Musically, we have found that in some cases, the relationship between signal A and signal B can become much

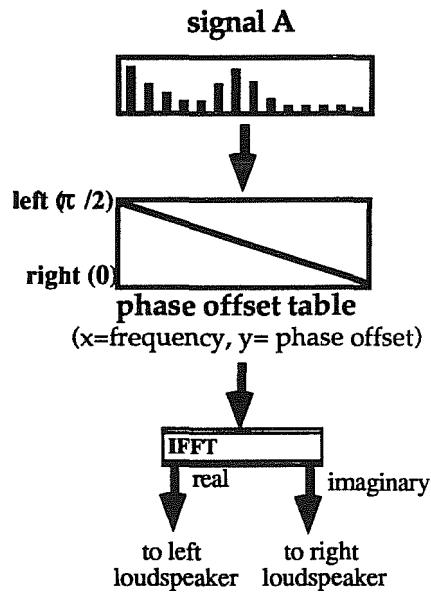
more unified if certain parameters of signal B are used to control signal A. In other words, real-time continuous control parameters can be derived from signal B and used to control signal A. Pitch and envelope following of signal B can yield expressive information which can be used to control the pitch and the intensity of frequency modulation of signal A. *see following figure:*



Frequency dependent spatialization

In the spectral domain, the phases of a given signal's frequency components can be independently rotated in order to change the component's energy distribution in the real and imaginary part of the output signal. Since the real and imaginary parts of the IFFT's output can be assigned to separate output channels, which are in turn connected to different loud-speakers, it is possible to control a given frequency's energy level in each loud-speaker using phase rotation. The user interface of this application permits users to graphically or algorithmically

specify the "panning" (phase offset) for up to 512 frequency components. *see following figure:*



Future Directions

The authors are currently working on alternative methods of sampling that operate in the spectral domain. Many interesting techniques for sound manipulation in this domain are proposed by the phase vocoder [7, 8]. Along with the possibility of modifying a sound's spectrum and duration independently, we would like to perform transposition independent of the spectral envelope (formant structure), thus allowing one to change the pitch of a sound without seriously altering its timbral quality.

Conclusion

With the arrival of the real-time FFT/IFFT in flexible, relatively general, and easily programmable DSP/control environments such as Max, non-engineers may begin to explore new possibilities in signal processing. Real-time convolution can be quite straightforward and is a powerful tool for transforming sounds. The flexibility with which spectral transformations can be done is appealing. Our DSP configuration is fairly simple, and changes to its topology and parameters can be made quickly. Control signals resulting from detection and tracking of musical parameters offer composers and performers a rich palette of possibilities lending themselves equally well to studio and live performance applications.

Acknowledgements.

The authors would like to thank Miller Puckette, Stefan Bilbao, and Philippe Depalle for their invaluable technical and musical insights.

References.

[1] Lindemann, E., Starkier, M., and Dechelle, F. 1991. "The Architecture of the IRCAM Music Workstation." Computer

Music Journal 15(3), pp. 41-49.

[2] Puckette, M. 1988. "The Patcher." In C. Lischka and J. Fritsch, eds. *Proceedings of the 1988 International Computer Music Conference*. San Francisco: International Computer Music Association.

[3] Puckette, M., 1991. "FTS: A Real-time Monitor for Multiprocessor Music Synthesis." Music Conference. San Francisco: Computer Music Association, pp. 420-429.

[4] Haddad R, and Parsons,T 1991, "Digital Signal Processing, Theory, Applications and Hardware", *Computer Science Press* (ISBN 0-7167-8206-5)

[5] Gordon, J and Strawn J, 1987. "An introduction to the phase vocoder", *Proceedings, CCRMA*, Department of Music, Stanford University, February 1987.

[6] Chowning, J. 1973. "The Synthesis of Complex Audio Spectra by means of Frequency Modulation" *Journal of the Acoustical Society of America* 21(7), pp. 526-534.

[7] Dolson, M. 1986. "The phase vocoder: a tutorial", *Computer Music Journal*, 10(4), Winter 1986

[8] Nieberle, R and Warstat, M 1992. "Implementation of an analysis/synthesis system on a DSP56001 for general purpose sound processing". *Proceedings of the 1992 International Computer Music Conference*. San Jose: International Computer Music Association.

Capitolo 6

WORKSTATION MUSICALI

Sezione 6a

*Stazione di Lavoro
Musicale Intelligente*

The cognitive level of the Intelligent Musical Workstation

Antonio Camurri

DIST - University of Genova
Computer Music and AI Labs
Via Opera Pia 11/A -16145 Genova
e-mail: music@dist.dist.unige.it

The *Intelligent Musical Workstation (IMW)* is a computer environment for computer music production and scientific research. It has been developed by the laboratories LIM-DSI of the University of Milan and DIST Computer Music and AI Labs of the University of Genoa in the framework of the project LRC C4: MUSIC, *Stazione di lavoro musicale intelligente - Intelligent Musical Workstation (IMW)*, Progetto C: Sistemi avanzati di produttività individuale, Sottoprogetto 7: Sistemi di supporto al lavoro intellettuale, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Consiglio Nazionale delle Ricerche (CNR).

The global architecture and the main tools are described in [3]. Here we focus on the cognitive level of the IMW, that is, on the high-level modules for sound and music processing based on Artificial Intelligence (AI), developed at DIST Computer Music and AI labs.

DIST-IMW Software Modules
The main DIST-IMW modules are the following:

- *XpetreX Development Tool* (X-windows PETRi nets EXecutor): a High-Level Petri nets-based tool for music and multimedia modeling and execution [1];
- *WinProcne/HARP* (WINdows PROlog tool Combining logic and semantic NEts for Hybrid Action Representation and Planning): a hybrid AI tool for music and multimedia composition [2, 4, 6];
- Tools based on artificial neural networks:
 - *ENA* (Experimental Neural Accompanist): real-time chords accompaniment of a single melody based on back-propagation [7];
 - *SOUL* (Self-Organizing Universal Listener): sound classification based on Kohonen neural networks [8].

XpetreX Development Tool

It is a Petri net-based software tool that allows users to model and execute music and multimedia objects by means of High-Level (Colored) Timed Petri nets. Its main features include:

- Colored Timed Petri net model;
- Dynamic binding of external C code associated to *transitions*;

- Object oriented extensions to Petri nets;
- Efficient Petri nets execution kernel, derived from the Petrex system.

XpetreX Development Tool is characterized by an advanced

graphical user interface which allows an easy editing/browsing of Petri net models, as it is sketched in figures 1 and 2. The system has been implemented in C language under Unix/X-Windows/Motif.

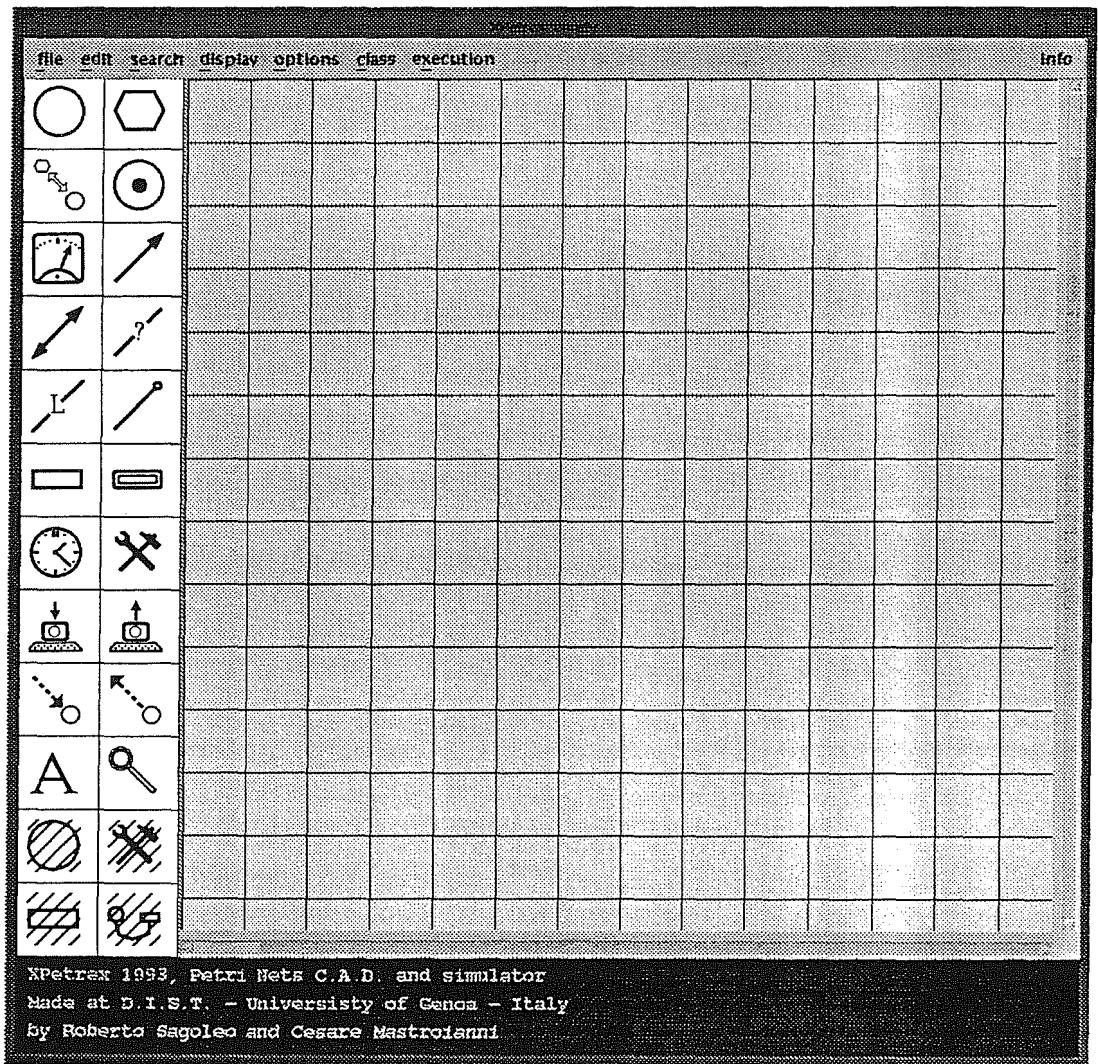


Figure 1: The XpetreX Editor/Browser main menu

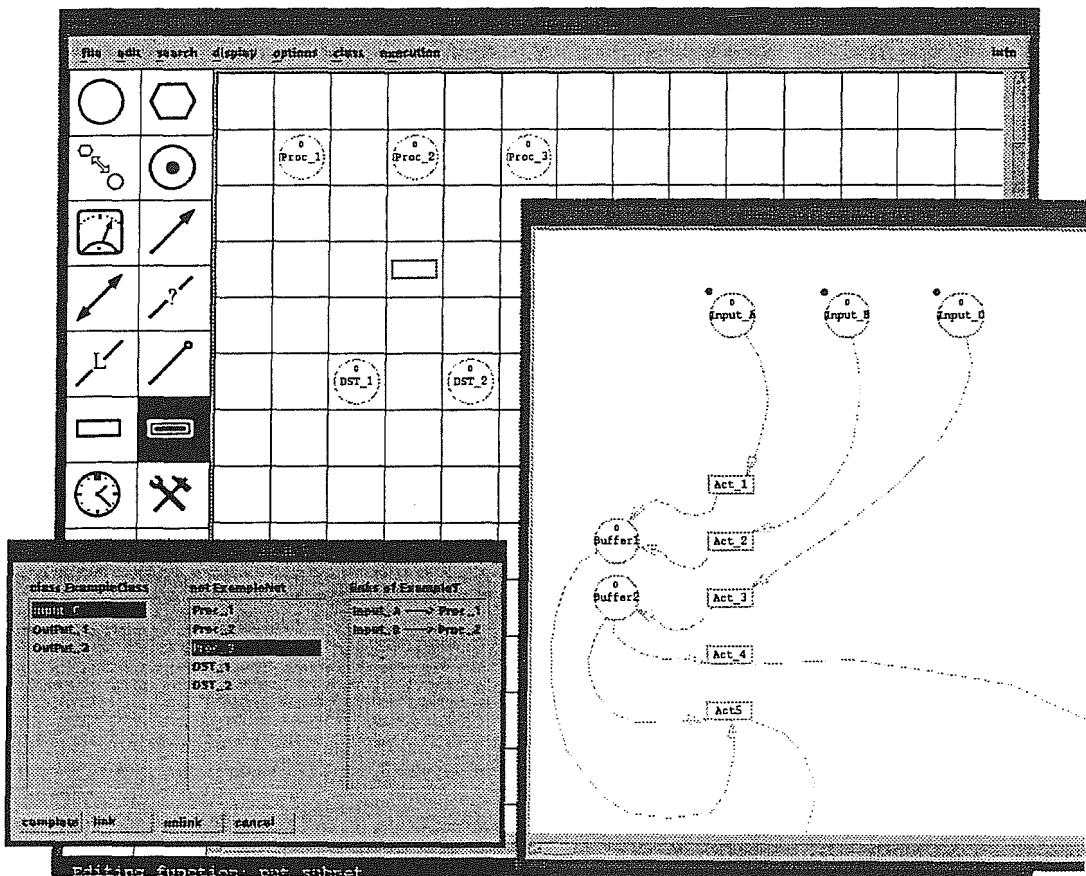


Figure 2: Introducing an instance of a Petri net class (small window on the right) as a refinement of a transition (main window). I/O places in the instance are assigned by the user to suitable places in the net in the main window.

WinProcne/HARP

WinProcne/HARP is a system for music and multimedia composition, based on a hybrid knowledge representation and reasoning architecture. It is based on the integration of different formalisms, able to manage the

different nature and levels of music and multimedia objects, from the symbolic, *knowledge level*, dealing with multi-level, abstract representations and reasoning mechanisms, to the *physical, analogical level*, dealing with low-level processes and

signals (including metaphors like *force fields*, *abstract potentials*). The *knowledge level* is realized by integrating a semantic network language of the family of KL-ONE and Krypton with a temporal logic language and production rules. The *analogical level* is modeled as an active memory based on metaphors (the AIM - active isomorphic memory), implemented as a concurrent object-oriented system, similar to Hewitt Actors system.

Our system has been designed for computer assisted composition and analysis, and for a particular, generalized 'multimedia' domain: a theatrical automation project, where the system is delegated to manage and integrate sound, music, and three-dimensional computer animations of humanoid

figures interacting with real actors on stage.

A detailed description of this system can be found in [2, 4, 6], and in a paper in preparation. An example application is shown elsewhere in these proceedings (see the paper by Massucco, Mercurio, and Palmieri).

Figure 3 shows system architecture. Figure 4 gives an idea of the high-level graphical user interface of the system.

The system is implemented in C++ and Prolog. Two versions are available: the former runs under Windows 3.1 (Microsoft Visual C++ and Arity Prolog 6.x), the latter under Silicon Graphics Indigo R4000, under Unix/X-Windows/Motif (native C++ and Sicstus Prolog compilers).

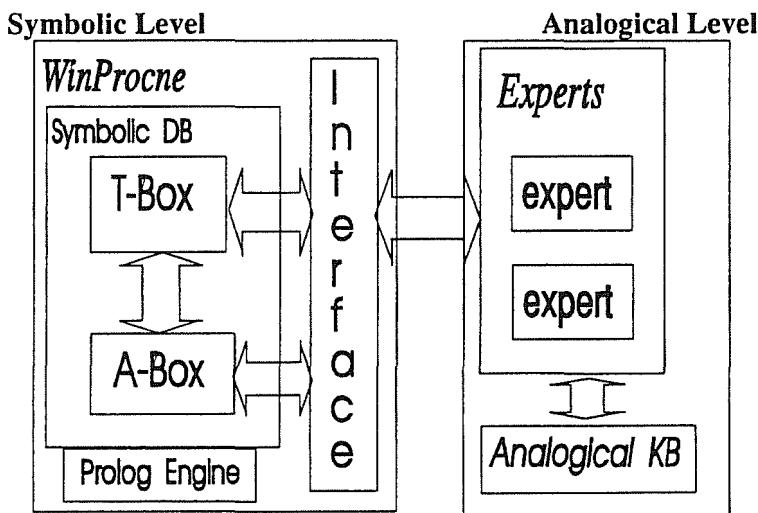


Figure 3: WinProcne/HARP system architecture

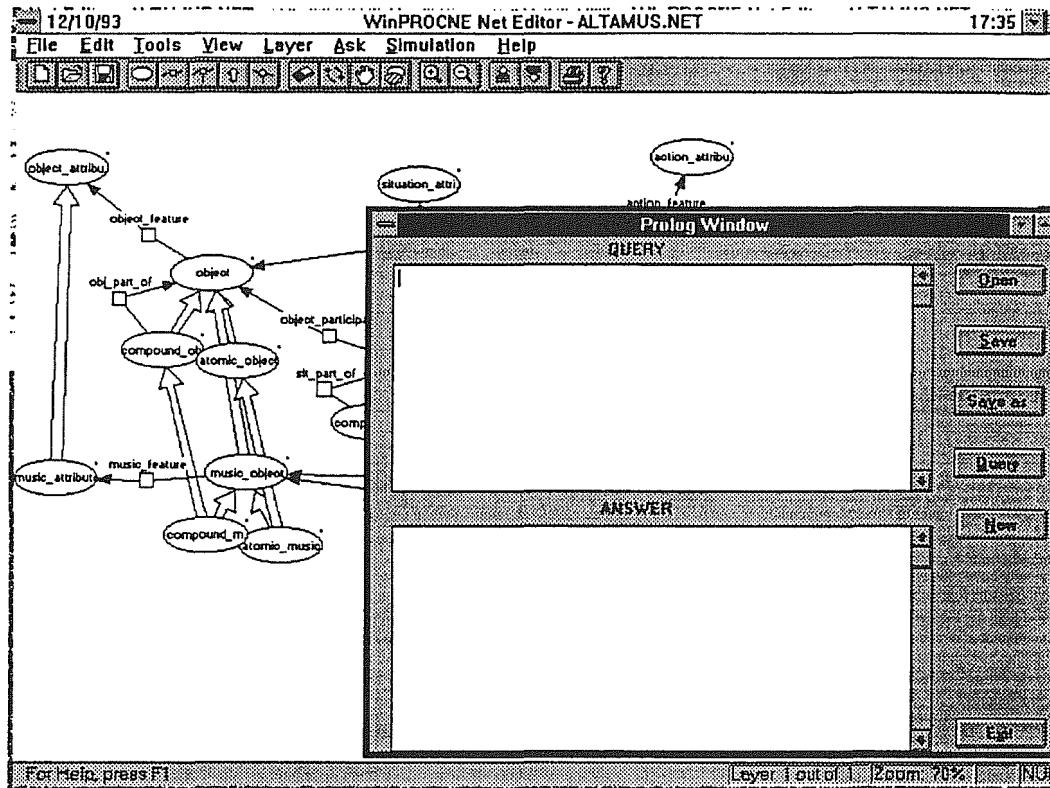


Figure 4: WinProcne/HARP graphical interface

DIST IMW Users

XpetreX has been extended for experimental use also in industrial set ups (FMS).

WinProcne/HARP is currently used by composers. The system has been utilized by Giuliano Palmieri (composer) and Mario Jorio (director) in a theatrical/musical event at Palazzo Ducale (Genova), and in a concert at Villa Arconati (Milano). A multimedia theatrical event at *Theatre H.O.P. Altrove*, Genova, directed by Jorio and Palmieri, in

which the system generates and integrates in real-time humanoid figure animations and computer music, is in preparation.

WinProcne/HARP is also experimented in collaboration with Marc Leman (University of Ghent) in a tonal centers recognition research problem [5].

Acknowledgments

XpetreX graphical editor/browser has been implemented by Paolo Franchi, Cesare Mastroianni and Roberto Sagoleo.

WinProcne/HARP version for Windows 3.1 has been implemented by Alessandro Catorcini, Carlo Innocenti, Alberto Massari, Alex Massucco, Marco Mercurio. The SGI implementation is by Stefano Andorno and Nicola Serini. Important contributes to the cognitive model and the design are due to Renato Zaccaria and Marcello Frixione.

DIST-IMW Bibliography

- [1] A.Camurri, G.Haus, and R.Zaccaria, "Describing and performing musical processes", *Interface*, Vol.15, No.1, 1986, pp.1-23, Swets & Zeitlinger, Lisse, The Netherlands. A revised version is included in P.Morasso and V.Tagliasco (Eds.), *Human Movement Understanding*, North Holland, 1986.
- [2] A.Camurri, C.Canepa, M.Frixione, and R.Zaccaria, "HARP: A System for Intelligent Composer's Assistance", *IEEE COMPUTER*, Vol.24, No.7, July 1991, pp.64-67. An extended version is included in D. Baggi (Ed.), *Readings in Computer Generated Music*, IEEE Computer Society Press, 1992.
- [3] A.Camurri, and G.Haus, "Architettura e ambienti operativi della Stazione di Lavoro Musicale Intelligente", Proc. *IX Colloquium on Musical Informatics*, Genoa, 13-16 November 1991, AIMI, and DIST - University of Genova.
- [4] A.Camurri, M.Frixione, C.Innocenti, and R.Zaccaria, "A model of representation and communication of music and multimedia knowledge", Proc. *ECAI-92*, Wien, 1992.
- [5] A.Camurri, and M.Leman, "Hybrid Representation of Music Knowledge - A case study on the automatic recognition of tone centers", Proc. *International Workshop on Models and Representations of Musical Signals*, Capri, 5-7 October 1992, University of Napoli, and AIMI.
- [6] A.Camurri, C.Innocenti, and C.Massucco, "A Multi-Paradigm Software Environment for the Real-Time Processing of Sound, Music, and Multimedia", *Knowledge-Based Systems*, Butterworth-Heinemann, 1993 (Forthcoming).
- [7] A.Camurri, M.Capocaccia, and R.Zaccaria. "Experimental Neural Accompanist (ENA)". Proceedings *International Neural Networks Conference INNC-90*, Paris, Kluwer Academic Publisher, 1990.
- [8] U.Bertelli, C.Bima, A.Camurri, L.Cattaneo, P.Jacono, P.Podesta` , R.Zaccaria. "Progetto SOUL: Un sensore acustico intelligente adattivo per il riconoscimento di sorgenti sonore", Proceedings *IX Colloquio Informatica Musicale*, AIMI and DIST Universita` di Genova, 1991.

Toward a cognitive model for the representation and reasoning on music and multimedia knowledge

A.Camurri, A.Catorcini, M.Frixione, C.Innocenti, A.Massari, R.Zaccaria

*DIST - Università di Genova
Laboratorio di Informatica Musicale
Via Opera Pia 11/A -16145 Genova
e-mail: music@dist.dist.unige.it*

Abstract: This paper introduces a cognitive model for the representation and real-time processing of music and multimedia, based on artificial intelligence (AI) techniques. The paper discusses some basic issues on the requirements of AI-based computer music systems. Then, our proposal of a representation scheme - at the basis of an implemented system, called *WinProcne/HARP* - is introduced.

1. Requirements of music and multimedia AI-based systems

Systems for sound and music processing are generally based on tools and programming languages designed for low-level manipulation of music scores and composition algorithms (Loy and Abbott, 1985). *Music V* (Mathews, 1969) and *cmusic* (Moore, 1990) are two well-known examples of such a class of traditional systems. More recent languages, such as *fugue* (Dannenberg, 1991), provide more elegant formalizations, e.g., aiming at the unification of the score and the orchestra languages into one language, and supporting some

abstraction mechanisms and functional programming to make easier the design process of composition algorithms. Other important steps in this direction are the introduction of object-oriented techniques (Pope, 1991), and other extensions to allow visual programming capabilities, as in *MAX* and *Edit20/MARS*, or to integrate hypermedia and multimedia objects in the music language, as in *SmOKE* (Pope, 1992). However, these are programming languages, including in their design more or less special purpose features for music processing. They generally are low-level tools, since often the composer is forced to work (program) at the level of MIDI messages: these languages have a very poor idea of music, and, above all, it is very difficult to abstract to higher level representations since these languages do not support high-level tasks. From a cognitive science viewpoint, these tools do not make any assumption, for example, on the problems of modeling human cognitive and perceptual skills. These problems are the main focus of the related research field on cognitive science and music (see for example Howell, West, Cross, 1991;

Krumhansl, 1990; Leman, 1989, 1992; Todd, 1992). In this scenario, research in AI and music plays a significant role: on one hand, research efforts are directed toward the definition of higher-level models for the manipulation of music in multi-media information systems¹; on the other hand, the research focuses on the definition of 'intelligent' systems for music processing, with the aim of both a deeper understanding of the perceptual and cognitive aspects of the 'musical mind', and to provide composers, musicologists, and researchers in general with more powerful, higher-level computer languages and tools. For example, significant studies are currently in progress on several areas: real-time performance and interpretation (De Poli, Irone, and Vidolin, 1990; Bresin, De Poli, and Vidolin, 1991), computer assisted music composition (Baird et al., 1990; Courtot, 1992; Ames, 1992), listening (Leman, 1991), tutoring (Dannenberg et al., 1990), music

analysis and musicological applications (Howell, West, and Cross, 1991).

The aim of our work is both to develop and experiment a high-level representation scheme for music and multimedia knowledge, based on AI techniques, and to implement a system for the "intelligent" assistance of users (including composers and musicologists).

In this paper, we focus on the cognitive model at the basis of our AI-based system. Before introducing our model, we discuss some basic issues at the basis of this new generation of systems.

The first issue deals with 'how' to approach AI for music research and applications: our conviction is that research should be grounded on a *bottom-up approach to AI*, starting from the psycho-physical, perceptual and cognitive findings. In other words, we deem that realistic, and therefore useful, AI models for music processing cannot be simply grounded upon symbolic, abstract entities, such as, for example, the starting notions of a textbook on the standard music notation. It is important to start from the psycho-physical, signal levels at which the human auditory system operates: many fundamental aspects of music understanding have their roots in its structure and behaviour. This implies that many music concepts often given as starting axioms in several logic-based music languages, should be considered, as a matter of fact, as the emerging result of the natural

¹In these last years, both researchers and industries have shown a growing interest in such kind of systems: in fields such as user-interface design, data and knowledge base management systems, the integration of capabilities encompassing graphic, animation, voice, sound and music into a flexible, high-level interactive multi-media computing system is one of the major goals. In the human-computer interaction field, important aspects regard the spatial and temporal management and processing of integrated multi-media objects.

processing of the auditory system. For example, the concept of *tonality* and the theory of the circle of fifths can be defined symbolically as a series of axioms in a sort of 'abstract' music knowledge base (KB); in an alternative approach, they have been demonstrated as an emergent structure derived from models of the human ear (Leman, 1992). In (Camurri and Leman, 1992) such an approach is faced in the framework of our representation architecture. This bottom-up approach is uncommon in AI-based systems, and this is one of the basic source of their failure or scarce utility, even from a theoretical point of view, since their reasoning models lack (part of) the semantics of the objects represented. Other examples of bottom-up approaches starting from the signal, perceptual level of representation can be found in (De Poli, Piccialli, and Roads, 1991), (Leman, 1992), and (McAngus Todd, 1992).

The problem discussed above can be put in strict relation, from an AI viewpoint, with the well-known problems of *symbol-grounding* and *situatedness* (Harnad, 1990; Kirsh, 1991; Steels, 1991).

Other important issues are the following: we need formalisms able to manage the different levels of *abstraction* of music and multimedia objects, from the *symbolic*, abstract representations (e.g., the voices of a canon) to the physical, *analogical* representations, like the signal perceived by the human ear. Furthermore, such formalisms should

allow composers, musicologists to define and *reason on plans*, strategies, on the basis given requirements, providing both formal and informal analysis capabilities for inspecting the objects represented.

Next, given the complexity of the problem domain, such systems should be characterized by integrated representations, each dealing with particular aspects: *combining different, complementary representations* is therefore a key issue in this kind of representation and reasoning architectures (a multi-paradigm approach, from the point of view of the computational mechanism). For example, taxonomic representations in terms of semantic networks (Brachman and Schmolze, 1985) are appropriate for inheritance and classification inference mechanisms; production rules are appropriate for representing logical implications; reasoning on actions and plans requires still further mechanisms, such as *analogical reasoning*, i.e., reasoning on the domain by means of "metaphorical" representations. In fact, metaphors are widely used in music, e.g., referring to the real world dynamics or kinematics (see for example Camurri, Morasso, Tagliasco, and Zaccaria, 1986): the metaphor of navigation in a space of *force* or *potential fields* is particularly useful, as it is discussed in section 2. From a cognitive point of view, this is the problem of representing *music imagery* and *mental models* of music knowledge. For example, the research of

McAngus Todd (1992) and Leman (1992) is based on the allusion of musical expression and perception to physical motion, including concepts of energy and mass: musical phrasing has its origin in the kinematic and dynamic variations involved in single motor actions. This is grounded on the psycho-physical structure of the human auditory system.

In this short overview of some of the crucial issues on music representation, it is interesting to recall the three constraints discussed in (Courtot, 1992) in the framework of the definition of the role of composition assisted environments: "*1)...assisting the composer to precisely formalize the syntax of musical structure. 2) ...handle the conceptual level of composers. 3) ...allow a composer to add new musical composition programs, without any skill in the particular programming language chosen for the system*" (p.191). The last point tackles a fundamental issue, regarding the *learning capabilities*, i.e., how to automatically update the system's knowledge (e.g., new analysis data, new composition strategies), for example by means of generalizations processes starting from examples presented by the user. The solutions proposed in literature, such as the purely symbolic approaches (Courtot, 1992; Widmer, 1992), and the learning systems based on neural networks, are interesting early attempts in this direction.

All the previous issues has been discussed in the context of music knowledge. One of our main goals is to *integrate both music and multimedia knowledge in a single representation architecture*. Let us discuss shortly what we mean as *multimedia knowledge*. Let us consider the two following scenarios: (i) a multimedia theatrical automation machine: a system delegated to manage and integrate sound, music, and either three-dimensional computer animations of humanoid figures (e.g., dance movements) or the movement of a real autonomous robot in a theatre stage (e.g., a real vehicle on wheels, equipped with on-board acoustic, proximity, infrared, and possibly other sensors, an on-board computer possibly connected with a more powerful remote computer via a radio link): such an 'agent' should be able to move, navigate, react to events happening on stage (e.g., actions performed by the actors), hear, and possibly execute musical tasks; (ii) a museal framework: an autonomous robot, very similar in its architecture to the theatrical machine, operates in real time in a museum exhibition, and is able to welcome, entertain, guide, and instruct visitors.

These two examples give an idea of what kind of "multimedia" knowledge we refer to. The basic objectives here are: (i) *integration* of different modalities and competences; (ii) *high level, multi-modal capabilities of interaction*

with the real world: the system should build up and reason on a realistic model of what is happening on stage, e.g., for deciding how to interpret or generate a music object, far more than a simple "triggering" mechanism. Interaction therefore means a deeper mechanism than a simple temporal synchronization of chunks of music and graphics/animation data.

2. The Cognitive Architecture

Our cognitive model finds its roots and motivations in the crucial issues discussed in the previous section. It is a general, abstract cognitive model for an autonomous agent exhibiting integrated music and multimedia competence. In this context, we refer to "multimedia knowledge" as the knowledge necessary to an autonomous agent for interacting with the real world, and exhibiting skills like those required in the theatrical and museal projects described in the previous section.

In figure 1, our cognitive model architecture is depicted. The three boxes are different "active storage" elements for knowledge representation and reasoning. Arrows indicate information exchanges during planning/acting.

The *active isomorphic memory* (AIM) is a component based on the analogical kind of representation mentioned above. The *symbolic memory* contains the symbolic representations that constitute the "high level" knowledge of the system. The AIM can either drive (plan and execute) external, real

actions on the world (e.g., drive a sound device, or a robot movement), or carry out actions in a *simulative model* of the world, or both. Actions and simulations can be triggered by the content of the symbolic memory, and, in turn, can generate new symbolic information. Moreover, actions and simulations in the AIM have a corresponding symbolic representation in the symbolic memory; the AIM executes performances/simulations and/or actions, described in the symbolic memory, and returns to it the relevant aspects *discovered* during the execution. The three components of figure 1 can be sketched as follows:

1. LTM (Long Term Memory): it is the permanent, "encyclopedic" storage of general music and multimedia knowledge;
2. STM (Short Term Memory): it is the actual "context", or "reasoning horizon", regarding the state of the affairs of the world and the problems actually faced by the agent;
3. AIM (Active Isomorphic Memory): it is an "active" (capable of generating and perceiving changes) world model, where a double mapping exists between perceptual data / motor commands and internal data, and between some perceptual / motor processes and internal "analogical models" of them.

Arrows LTM ↔ STM in figure 1 indicate the information flow that keeps the context updated. As long as the horizon of the action evolves, new long term information is instantiated in the STM to build a local description of the state of the

affairs. For example, in the process of composing a piece in a classic *sonata* form, the context related to the knowledge on the *exposition* part of the *sonata* will only be initially instantiated by the system. This instantiation of the symbolic description of the *exposition* (e.g., the definition of the two contrasting themes and their evolution) is considered short term because its existence is bounded to the current composer's task. The subsequent parts of the sonata form will be considered for instantiation into the context only when necessary. Once instantiated, the context is completed with the corresponding information in the AIM (e.g., the low-level, operational semantics of the instanced symbols, as it will be described in the following subsection). The mechanism of context definition is very important, and is one of the key issues of our architecture; planning/reasoning systems which do not *ground* the representation on some kind of context mechanism fail to be realistic (McCarthy, personal communication).

The STM includes the information concerning the specific events represented in the AIM. In other words, the STM represents a single context, the bounded horizon of the action represented in the AIM: it is a one-to-one symbolic representation of the entities and of the events in the AIM. The arrows in figure 1 schematise the relations between the various components: a specific context is created starting from the

knowledge in the LTM (arrow LTM → STM); starting from this context, a model of the action is generated (arrow STM → AIM); the AIM feeds back its "discoveries" to the STM (by simulations, performance or analogical reasoning based on metaphors like navigation in *force fields*), and asks for new information (arrow STM ← AIM). The STM in its turn asks for relevant information the LTM and possibly updates it if new long-term knowledge is discovered (arrow LTM ← STM).

The arrow STM ← AIM indicates the sensing path. Apart from low level sensing, managed directly in the AIM, it carries signals which do not update, but rather complete the representations in the two upper memories, or are requested by reasoning processes. On the contrary, the arrow LTM ← STM is an information path which provides learning from acting.

Our model stands on the assumption that every action execution always involves a certain amount of planning: it may range from pure adaptation up to a complete planning process (e.g., discovering a sequence of sub-actions; designing postures for a redundant complex robot like a humanoid model). We expect the agent to *learn from experience*, to re-use conclusions and results from previous experiments or reasoning processes. Learning has basically two components: extracting relevant information, and classifying it. The former requires an "attention"

mechanism, for selecting the pieces of knowledge which are relevant. The latter requires a suitable architecture necessarily centered around the "storage" of knowledge. Unlike other architectures, our model is suited for upgrading its knowledge. The AIM → STM arrow indicates the information flow due to experience, a *cognitive feedback* from the state of the affairs of the action towards the two symbolic memory boxes, to store information valid in the actual context, or valid universally.

The possibility of learning is mainly due to the choice of the cognitive scheme for representing actions which is as uniform as possible, and constructed around the metaphor of "showing by examples" actions, skills and goals. The semantics of the action representation scheme is the way in which the AIM works.

2.1 The Active Isomorphic Memory

The AIM is basically a complete, isomorphic model of the world. It is *isomorphic* because, in a sense, it has "the same shape" of possible or actual situations of the external world: changing the value of a variable in the AIM is equivalent to change the value of the corresponding variable in the (real or hypothetic) represented situation. The AIM is analogous to a *mental model* in the sense of Johnson-Laird (1983): it can be a model of the real world used to figure out the possible outcomes of a given action. This

idea is also inspired to the regions of the brain which are mapped onto relevant sensor-motor parts of the body (see for example Sparks, 1987). The importance of the metaphor of *mental models* in music has been already pointed out (Krumhansl, 1990; Leman, 1991). The mental model we refer to is more general, since our goal is to represent both music and multimedia: it is a common representation framework for both models of actions in a three-dimensional (simulated or real) world (e.g., choreographic situations represented by the system), and *music imagery*, based on psychoacoustics and analogical reasoning (see for example McAngus Todd, 1992).

One important role of the AIM is that of a motion generator based on reference "pictures", or "snapshots" of parts of the world. We call *icons* these multi-dimensional snapshots stored in the AIM. According to the representation context, icons can refer to different kinds of knowledge:

- (i) geometrical descriptions of situations in a model of the world (they can be analogical representations, i.e., geometrical metaphors of a different domain, e.g., music): simple measures on a picture can reveal that, say, object A is "on the left" with respect to object B. This is useful for *spatial and geometrical reasoning* purposes;
- (ii) landscapes of energy, force fields, as a powerful metaphor for

reasoning on actions and plans. Here we have an *analogical reasoning* mechanism, based on measurements and simulation.

Both previous reasoning mechanisms are complementary to typical symbolic deductive systems.

The AIM is therefore *active*, since it generates the suitable motor activities to make icons become "true" (or, better, to change the world to fit the icon descriptions).

Icons can be "forbidden snapshots" as well, as to say, examples of states that must not be reached. We call them informally *repelling icons* (or *repellors*), while *attractive icons* are 'good' icons, representing states of the world to be reached. This framework recall the basic mechanism at the basis of the AIM: navigation in *force fields*.

Another role of the AIM is purely simulative: in this case, *icons* are used as energy shapes which can be followed by means of algorithms simulating attraction forces on a moving point, e.g., to reach a minimum of energy or a given goal position.

Two important properties of the AIM are to be kept into account: (i) the AIM can always represent the evolution of a single state of affairs at once. That is, no general information can be represented in it; (ii) the AIM contains always complete knowledge with respect to the relevant aspects of the represented state of affairs. No incomplete piece of information can

be represented in it. For example, with respect to disjunction, it is impossible to represent in the AIM that things are so or so, without saying exactly which is the case.

These two properties are consequences of the "iconic" nature of its representations. On the contrary, the symbolic memory is more expressive from this point of view. The possibility of explicitly expressing generalizations, and of representing incomplete pieces of knowledge, and of reasoning with them, is one of the main reasons to introduce this symbolic level of representation.

An important advantage of an *iconic* scheme is that motion processes (which "interpolate" from one icon to another, like graphic interpolation between the so called *key postures* in the cartoons terminology) can use *composable* generation processes. The impossibility of composing actions in time and space (by adding, superimposing partially or totally the effects of different actions, without consistency problems) is one of the weak points when using purely symbolic, logic-based approaches. In our architecture, two or more icons can be simultaneously active, and the resulting action comes from the superimposition of the different, independent generation processes into the AIM, which integrates the effects of the different processes in the same way the world does. This also has a good evidence and expressive strength: a complex action can be captured, at any

instant, as a set of active icons in the AIM to be satisfied, and a set of repulsive situations to keep away from.

We have introduced icons as the driving mechanism for motion processes: this constitutes only one half of their role in the AIM. In a dual way, icons can be generated by perceptual processes. See for example the iconic representation of the tonal circle of fifth in (Leman, 1991), where the iconic representation is implemented using the Kohonen's self-organizing neural networks (Kohonen, 1984).

Such perceived snapshots are the basis for decisions about what to do next in a plan; in other words, they allow the action representation to be "situated" (Brooks, 1991).

In the same way, more complex recognition processes, involving explicit reasoning using symbolic knowledge, can be carried out in the LTM and STM; the reasoning process, in this case, uses the situations fed back by the AIM.

Planning carried out in the AIM is a particular form of *metaphorical problem solving using simulation*, since it is based on the *abstract force fields* metaphor.

Summing up, the generating activity of the AIM ranges three distinct roles:

(i) **execution**, (no generative activity): the AIM is an interface between the symbolic entities in the context (STM) and the real world;

(ii) **simulation**, (low generative activity): to "substitute" (a mental model, disconnected from the real world) the world for reasoning, or to anticipate and foresee the behavior of (some portion of) the world;

(iii) **planning**, (high generative activity): to discover how to get to a desired snapshot and/or staying away from an unwanted situation, at different levels of abstraction.

Local minima of energy are the main sources of information returned to the STM and the LTM. At every local minimum, a repelling icon, and a corresponding symbol in the *symbolic memory*, can be generated. After such a new symbol is generated, the STM can be updated, and the action starts again. This cycle can be iterated while upgrading both the LTM and the STM. For example, after the first iteration, the representation of the action is composed by an initial situation, a final situation, and is now enriched by an intermediate "bad" situation, with additional space/time information. Examples of such process are discussed in later sections.

A generative scheme of motion based on force fields has good properties for being the basic reasoning mechanism in the AIM, since:

- force fields can be composed;
- they can integrate different levels of representation: kinematics, force, internal and real values;

- they use local representations;
- algorithms for exploring force fields are known; local minima (as well as other energy shapes) are source of information;
- force fields are a metaphor which is sufficiently general: it is used in several domains.

Finally, an interesting feature regarding the AIM is its internal simplicity. It has basically four components, depicted in figure 2: a *set of icons* (the current context), a *generative process*, a *minima detector*, and a *clock*. The AIM inputs icons (in the actual context), and gives back new icons, corresponding to the discovered relevant situations; output icons carry proper ordering information, a subset of complete timing information, for which an elementary clock is necessary.

Concluding, we have introduced a representation and reasoning framework based on the metaphor of *force fields*. These descriptions let the user think of and perform a set of actions in terms of the intuitive natural dynamics of navigation in attractor fields, as a promising alternative to production rules: given their *composable* generation processes, force fields can be used to easily model complex behaviours, otherwise very difficult to model as rules. Force fields also give a different viewpoint to the composer or the musicologist, and provide powerful manipulation primitives.

3. A computational model

From the point of view of the representation and reasoning architecture, the formal realization of the cognitive model described so far consists of a hybrid scheme, combining different formalisms, as it is shown in figure 3.

The *symbolic memory* is realized by the two following schemes, for representing different kinds of knowledge: (i) *multiple-inheritance semantic networks*, a formalism appropriate for defining terms and for describing objects and the taxonomic class/membership relations among them. This formalism has been extended to represent and reason on time, actions and plans; (ii) *production rules*, appropriate for representing logical implication. This integration of taxonomic representations with rules is similar to the CLASP system (Yen, Neches, and MacGregor, 1991).

The AIM is mainly based on a *situated action* model and on *reasoning on analogies*: the force field metaphor described in the previous section is its main inference engine.

From the point of view of the computational mechanisms, the system integrates three paradigms:

(i) a representation language of the family of *KL-ONE* (Brachman and Schmolze, 1985), extended with time management primitives for reasoning on time and actions; (ii) a rule-based system, operating on

terms defined in the terminological language, and with a full access to the Prolog language; (iii) the AIM component, implemented as a concurrent, object oriented system, similar to Hewitt's *actor system* (Agha and Hewitt, 1990), implemented in C++.

The low-level sound representation and data on real-time performance, as well as particular recognition and synthesis algorithms, are therefore represented in the system as classes (actors) in an object-oriented concurrent environment: they correspond (are "hooked") to terms in the Prolog KB. The AIM is therefore implemented as a network of *actors*: each *actor* is therefore a *class* hooked to an object in the symbolic level, which is the repository of high level entities, scores, composition rules, high-level descriptions and definitions in general. The activation of such a network of actors produces a simulation/execution in the AIM.

More details on the system, a new version of *WinProcne/HARP*, and on musical and multimedia applications can be found in (Camurri et al, in preparation) and elsewhere in these proceedings (Camurri, Giuffrida, Vercelli and Zaccaria, 1993; Massucco, Mercurio, and Palmieri, 1993).

References

G.Agha, C.Hewitt, "Actors: A Conceptual Foundation for Concurrent Object-Oriented Programming", in *Research Directions in Object-Oriented Programming*, B.Schrivier

and P.Wegner (Eds.), Cambridge, MIT Press, 1987.

C.Ames, "AI and Music", *Encyclopedia of Artificial Intelligence*, 2nd ed., 1992, New York: John Wiley and Sons.

C.Ames, "Quantifying Musical Merit", *Interface*, Vol.21, No.1, pp.53-93, 1992, Swets & Zeitlinger.

R.J.Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, 9, pp.171-216, 1985.

R.A.Brooks, "Intelligence without representation", *Artificial Intelligence*, Vol.47, no.1-3, 1991.

A.Camurri, C.Canepa, M. Frixione, and R. Zaccaria, "HARP: A framework and a system for intelligent composer's assistance", *IEEE COMPUTER*, Vol.24, No.7, pp.64-67, July 1991.

A.Camurri, M.Frixione, C.Innocenti, R.Zaccaria, "Representing and reasoning on music and multimedia knowledge", DIST Techn. Report, University of Genova, 1993.

A.Camurri, F.Giuffrida, G.Vercelli, and R.Zaccaria. "A system for real-time control of human models on stage", this volume.

A.Camurri, P.Morasso, V.Tagliasco, and R.Zaccaria, "Dance and movement notation", in P.Morasso and V.Tagliasco (Eds.), *Human Movement Understanding*, North Holland, 1986.

F.Courtot, "CARLA: Knowledge Acquisition and Induction for Computer Assisted Composition", *Interface*, Vol.21, pp.191-217, Swets&Zeitlinger.

R.Dannenberg, C.Lee Fraley, P.Velikonja, "Fugue: A Functional Language for Sound Synthesis", *IEEE*

- COMPUTER*, Vol. 24, No.7, July 1991, pp.36-41.
- G.De Poli, L.Irone, A.Vidolin, "Music score interpretation using a multi-level knowledge base", *Interface*, Vol.19, No.2-3, pp.137-146, 1990, Swets & Zeitlinger.
- G.De Poli, A.Piccialli, and C.Roads (Eds.), *Representations of Musical Signals*, MIT Press, 1991.
- S.Harnad, "The symbol grounding problem", *Physica D*, Vol.42, No.1-3, 1990.
- P.Howell, R.West, I.Cross (Eds.) *Representing Musical Structure*, Cognitive Science Series, Academic Press, 1991
- P.N.Johnson-Laird, *Mental models*. Cambridge University Press, Cambridge, 1983.
- T.Kohonen, *Self-organization and associative memory*, Berlin: Springer-Verlag, 1984.
- C.L.Krumhansl, *Cognitive foundations of musical pitch*, Oxford University Press, New York, 1990.
- M.Leman, "The ontogenesis of tonal semantics: results of a computer study". In P.Todd & G.Loy (eds.), *Music and connectionism*. Cambridge, MA : The MIT Press, 1991.
- M.Leman, "Tone context by pattern-integration over time". In D. Baggio, (Ed.), *Readings in computer generated music*, pp.117-137, IEEE Computer Society Press, 1992.
- G.Loy, and C.Abbott, "Programming Languages for Computer Music Synthesis, Performance and Composition", *ACM Computing Surveys*, Vol. 17, No. 2, June 1985, pp.235-265.
- C.Massucco, M.Mercurio, and G.Palmieri, "Real-time processing and performance using WinProcne/HARP", this volume.
- M.V.Mathews, *The Technology of Computer Music*, MIT Press, Boston, 1969.
- N.P. McAngus Todd, "The dynamics of dynamics: A model of musical expression", *J. Acoust. Soc. Am.*, Vol.91, No.6, June 1992, pp.3540-3550.
- F.R.Moore, *Elements of Computer Music*, Prentice Hall, Englewood Cliffs, N.J., 1990.
- S.T.Pope (Ed.), *The Well-Tempered Object*, 1991, MIT Press.
- S.T.Pope, "The Interim DynaPiano: An integrated computer tool and instrument for composers", *Computer Music Journal*, Vol.16, No.3, Fall 1992, MIT Press.
- S.T.Pope, "The SmOKE Music Representation, Description Language, and Interchange Format", Proc. *Intl. Workshop on Models and Representations of Musical Signals*, Capri 5-7 October 1992, AIMI (Italian Association on Musical Informatics), and Dipartimento di Scienze Fisiche, University of Napoli.
- J.Yen, R.Neches, and R.MacGregor, "CLASP: Integrating Term Subsumption Systems and Production Systems", *IEEE Transactions on Knowledge and Data Engineering*, Vol.3., No.1, pp.25-32, March 1993.

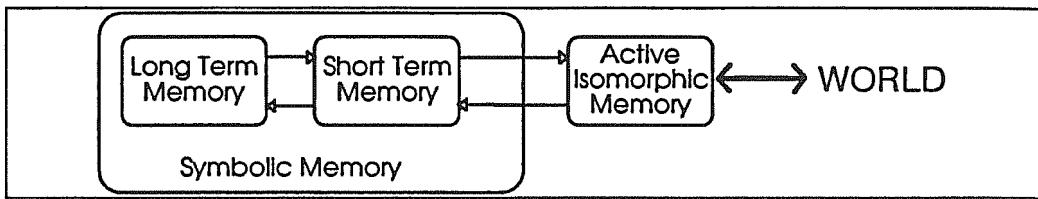


Figure 1. The general architecture of the cognitive model at the basis of the WinProcne/HARP system.

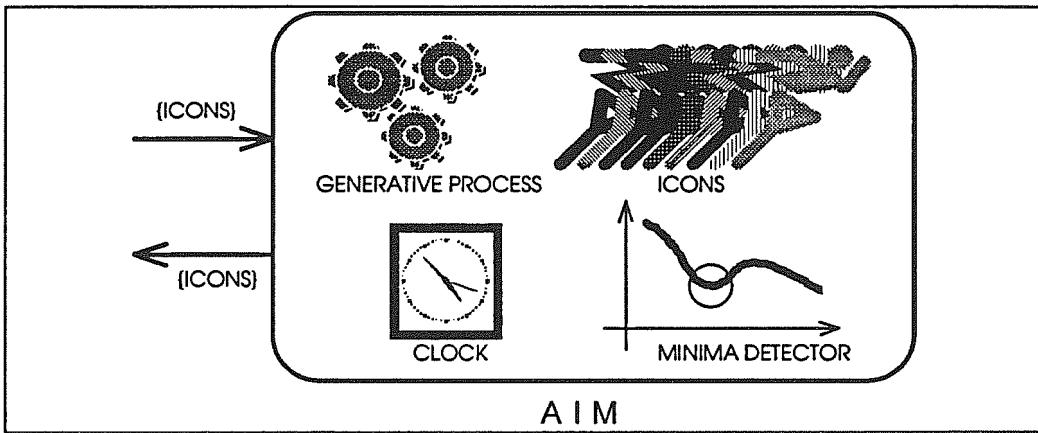


Figure 2. The structure of the Active Isomorphic Memory.

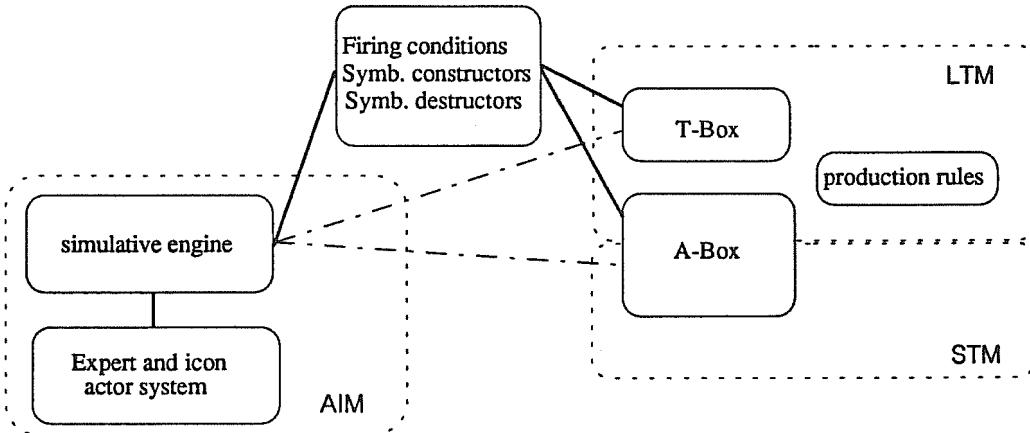


Figure 3. WinProcne/HARP system's architecture.

DUE AMBIENTI Sperimentali DEDICATI ALLA Sintesi LASY

Jacques Chareyron, Daniele Rizzi

L.I.M - D.S.I, Università di Milano

Via Comelico, 39

I-20135 Milano (Italy)

fax +39 2 55006.373

e-mail: music@imiucca.csi.unimi.it

Abstract

Sound synthesis with cellular automata (LASy) is an application of the cellular automata model to the digital signal processing field.

Two working environments for the implementation of the LASy algorithm have been implemented at LIM on two hardware set-up.

The LASy application uses a Macintosh computer equipped with an expansion card including a DSP and a DAC. The LASy application may be seen as a MIDI expander for LASy synthesis with accomodation for programming and building sounds.

The application allows the building and the grouping of the basic elements of the synthesis. The generated tone may be heard immediately by using whether the computer's keyboard or a MIDI-connected musical keyboard; the evolving waveform is displayed in real-time.

The performance section of the LASy application allows four voices polyphonic synthesis under MIDI control.

CellAut software runs on a NeXT computer suited for digital processing. It allows parametrical and graphical editing of the transition rule; changes take effect immediatly since the synthesis algorithm and the user interface handler are two concurrent processes.

The user gets control on a vaste array of other minor parameters, such as the kind and the frequency of the waveform; these parameters are useful in refining a single timbre or for exotic use of the application.

Another feature of the software is the "pitch tracking" mode: the signal coming from a microphone is sampled using the ADC and analyzed by a program running on the DSP. Pitch and power characteristics are extracted from the input, so the user may control the automaton's evolution and the corresponding timbre's synthesis with changes in the pitch and the tone of his voice.

This research has been supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

La tecnica di sintesi timbrica con automi cellulari (LASy) è una applicazione al campo della elaborazione dei segnali digitali del modello matematico degli automi cellulari, definiti da uno stato iniziale e da una funzione di transizione.

Riempiendo lo stato iniziale con i valori di una forma d'onda di partenza e applicando ricorsivamente la funzione di transizione, si ottiene una successione di stati. Il risultato può essere interpretato come la descrizione di un segnale digitale.

Due ambienti di lavoro per lo studio e l'utilizzo dell'algoritmo di sintesi LASy sono stati sviluppati al LIM su due diverse piattaforme hardware.

L'applicazione LASy utilizza un calcolatore Macintosh dotato di una scheda di espansione con DSP e convertitore digitale-analogico. Si tratta di una realizzazione dedicata sia alla sperimentazione che all'esecuzione musicale. I risultati della ricerca sul suono possono essere sfruttati per l'esecuzione musicale all'interno della stessa applicazione anche da parte di un musicista che non abbia familiarità con la tecnica di sintesi. In effetti la combinazione calcolatore-scheda DSP-applicazione LASy può essere vista come un expander MIDI dedicato alla sintesi LASy dotato di ampie capacità di programmazione e di costruzione del suono e di una libreria di suoni preprogrammati.

LASy dispone di moduli per la costruzione e la modifica degli elementi di base della sintesi (forme d'onde e regole di transizione). Questi elementi possono essere combinati in diversi modi e completati con l'aggiunta di inviluppi e modulazioni per formare uno "strumento" LASy. A ogni istante della fase di costruzione è possibile ascoltare il suono risultante agendo sulla tastiera alfanumerica del calcolatore e il mouse, o su una tastiera musicale collegata via MIDI. L'evoluzione della forma d'onda corrispondente viene visualizzata in tempo reale sullo schermo collegato al calcolatore.

La costruzione di uno "strumento" LASy può essere condotta a diversi livelli a seconda della familiarità dell'utente con il meccanismo della sintesi. Il livello "zero" consta nella semplice scelta di uno degli strumenti forniti insieme all'applicazione. Al primo livello l'utente costruisce nuovi suoni combinando gli elementi di base della sintesi (forme d'onde e regole di transizione) forniti a questo scopo. Al secondo livello l'utente provvede alla costruzione degli elementi di base della sintesi. A questo scopo l'applicazione LASy dispone di una serie di algoritmi specializzati per ogni tipo di elemento. Al terzo livello si ha un controllo assoluto sulla costruzione o la modifica "bit per bit" degli elementi della sintesi grazie a due moduli dedicati: un editor in modo "testo", per l'introduzione

diretta dei valori scelti, e un editor grafico con il quale si può "disegnare" la curva che rappresenta graficamente la tabella da elaborare. L'applicazione LASy dispone inoltre di una serie di operatori che applicano una trasformazione su una zona selezionata degli elementi della sintesi.

La sezione dell'applicazione LASy dedicata all'esecuzione musicale offre diversi modi di sintesi, fino ad un massimo di polifonia a quattro voci. La sintesi utilizza lo strumento corrente, anche in fase di costruzione o di modificazione. Nell'assenza di collegamenti MIDI, la tastiera alfanumerica del calcolatore può essere usata come una elementare tastiera musicale. Il collegamento ad una tastiera MIDI esterna (o a un sequencer) autorizza un controllo più completo degli elementi della sintesi. L'applicazione LASy è compatibile con il MIDI Manager del sistema Apple Macintosh. E' possibile fare girare LASy in background mentre risponde ai messaggi MIDI mandati da un'altra applicazione (tipo sequencer) che gira in primo piano sullo stesso elaboratore. LASy risponde ai messaggi MIDI di Note On, Note Off, Program Change, After Touch, Pitch Bend e due Control generici a scelta.

Come ambiente di ricerca, la versione attuale dell'applicazione LASy è completa e operativa. Potrà essere completata ulteriormente con altri algoritmi di co-

struzione. Come strumento musicale, l'applicazione è pienamente funzionante, ma sono ancora presenti difetti nella qualità del suono emesso rispetto a materiale di livello professionale. Una parte di questi difetti è legata ai limiti della base hardware adottata. Si sta lavorando per eliminare gli altri.

L'applicazione CellAut nasce come banco di prova per lo studio della tecnica di sintesi LASy. L'ambiente di sviluppo è consistito in un calcolatore NeXT, sia per i suoi requisiti hardware che per i tools di sviluppo a disposizione. Sulla piastra madre del calcolatore sono presenti sia i DAC sia un coprocessore dedicato DSP, a cui è stato aggiunto un campionatore esterno; è disponibile perciò tutto l'hardware necessario per l'analisi e la sintesi del segnale digitale. Inoltre, il sistema operativo offre una serie di tool che consentono di semplificare lo sviluppo dell'applicazione e di utilizzare efficacemente l'interfaccia utente grafica.

Il risultato di una sintesi LASy dipende da tre parametri principali: la forma d'onda iniziale, la funzione d'intorno $s()$ e la funzione $f()$, legate dalla relazione:

$$Y_n = f \circ s(Y_{n-p}, Y_{n-p+1}, Y_{n-p+2}),$$

dove $Y_{0...p-1}$ rappresenta la forma d'onda iniziale e $Y_{p...k}$, $k > p$ il segnale prodotto dalla sintesi. CellAut permette di intervenire sui tre parametri

principali, di ascoltare e visualizzare graficamente il risultato della sintesi, e di trasferirlo automaticamente verso altre applicazioni, ad esempio per produrre un sonogramma.

La forma d'onda iniziale $Y_{0..p-1}$ può essere scelta dall'utente tra un insieme di onde classiche, quali la sinusoidale o la quadrata, oppure come un frammento di un segnale campionato; in questo caso LASy viene utilizzata non come algoritmo di sintesi ma come filtro digitale.

La funzione d'intorno $s()$ ha forma:

$$s(Y_{n-p}, Y_{n-p+1}, Y_{n-p+2}) = a_0 \times Y_{n-p} + a_1 \times Y_{n-p+1} + a_2 \times Y_{n-p+2};$$

con $a_0, a_1, a_2 > 0$ e $a_0+a_1+a_2 < 1$. $s()$ definisce un filtro digitale di tipo passa-basso, le cui caratteristiche dipendono dai tre parametri a_0, a_1 e a_2 , modificabili tramite un pannello.

$f()$ è una funzione, in genere non lineare, che opera sul segnale filtrato da $s()$; ragioni di efficienza consigliano di precalcolare $f()$ in una tabella di ricerca, o lookup table, in modo da ridurre e uniformare il tempo di calcolo di ogni istanza della stessa funzione. Un effetto secondario è che l'utente si trova a disposizione sia una funzione parametrica, sia una tavola di valori visibili e modificabili graficamente. È possibile sia modificare $f()$ cambiando i suoi coefficienti, sia agendo con il mouse sul suo grafico. Grazie alle poche operazioni necessarie ad aggiornare

la tabella, l'operazione è eseguibile durante la sintesi, permettendo di ascoltare direttamente i suoi effetti sul suono generato. A causa della sua generalità l'effetto di $f()$ è solitamente difficile da stabilire a priori; i risultati migliori si ottengono con la funzione identità ($f(x) = x$) e ogni funzione "vicina". Allontanandosi da questo insieme, l'evoluzione risulta troppo veloce per poter essere apprezzata e porta in tempi molto ridotti ad un segnale nullo o ad un segnale composto da tutte le armoniche alla massima energia.

Per ottenere un controllo più preciso del suono prodotto, si è collegata l'uscita dell'algoritmo LASy ad un pitch detector. Esso monitorizza l'input rilevato da un microfono e ne stabilisce l'altezza, l'energia e se debba essere considerato vocale o meno; indi il programma decide se produrre il suono sintetizzato (nel caso di un input vocale), e ne cambia la frequenza per uguagliarla a quella rilevata dal microfono. L'altezza del suono prodotto viene perciò stabilita dalla voce stessa dell'utente, in una maniera più intuitiva e immediata della usuale interazione con la tastiera.

Bibliografia

J. Chareyron, "Digital Synthesis of Self-Modifying Waveforms by Means of Linear Automata", Computer Music Journal, Vol.14, N.4, 1990, pp.25-41.

PC-MUSIC - EVOLUZIONE DEL LINGUAGGIO CMUSIC PER AMBIENTE MS-DOS

Pietro Fischetti

DIST - Universita' di Genova
Laboratorio di Informatica Musicale
Via Opera Pia 11/A 16145 Genova

Abstract

This paper describes a set of programs for musical and digital signal processing called "PC-MUSIC". This Software has been implemented in C language, running on MS-DOS systems and is actually composed by: CMUSIC for MS-DOS, ENS (programs and library for Digital Signal Processing), GED (Graphic Editor), VMS (Virtual Memory System). The system is modular and easily transportable.

1. Introduzione

Il presente articolo illustra una raccolta di programmi per la composizione musicale, e l'elaborazione sonora sviluppati presso il laboratorio di Informatica Musicale dell'Universita' di Genova DIST appartenenti al modulo DIST.Music.Tool (DMTOOL) del progetto "Una Stazione di

Lavoro Musicale Intelligente" [1]. Viene presentato il trasporto del linguaggio CMUSIC da ambiente UNIX ad ambiente MS-DOS e una raccolta di programmi standard per l'elaborazione dei segnali sonori, infine viene descritta una libreria per la gestione virtuale a pagine della memoria di un PC-IBM. L'hardware utilizzato e' un comune PC-IBM in ambiente MS-DOS (Ver. 3.30 o sup.). Il software utilizzato e' il linguaggio MicroSoft C Ver. 5.10. Le restanti parti del modulo DMTOOL riguardano driver per la comunicazione tra PC, Campionatore AKAI S900 e MIDI Roland MPU-401 [6].

2. Descrizione del sistema

PC-MUSIC e' composto dai moduli Hw/Sw descritti in Figura1, in particolare:
la parte Hardware comprende un comune PC-IBM, schede AD/DA, eventuali schede DSP.

Il software e' composto da:

HDMUSIC: e' un driver configurabile per l'interfacciamento PC - schede AD/DA via DMA.

GED: Editor grafico di campioni sonori.

ENS: Raccolta di programmi per l'elaborazione numerica dei segnali.

VMS: gestore virtuale della memoria.

CMUSIC: compilatore.

3. I campioni sonori

I possibili formati dei campioni sonori utilizzati (in banda base) sono i seguenti:

FLOATSAM: dati binari reali a virgola mobile in singola precisione (6 digit) a 32 bit.

SHORTSAM: dati binari interi a 16 bit in complemento a 2.

Un file di campioni sonori puo' essere o meno preceduto da un'intestazione (header) che consiste in una sequenza di informazioni riguardanti principalmente il nome del file, il numero dei canali (mono, stereo, o quadrifonici) e la frequenza di campionamento. Viene fornita una libreria di funzioni per l'ingresso/uscita dei dati a basso o alto livello, e per il trattamento dell'header.

4. Il Driver HDMUSIC

HDMUSIC permette il trasferimento di campioni, (shortsam), tra il DMA del PC e la scheda AD/DA [5]. La frequenza di trasferimento puo' essere impostata da 8 a 84KHz (mono) e da 4 a 42KHz (stereo).

5. L'editor Grafico GED

GED visualizza file sonori su un video grafico ad alta definizione (VGA). L'utilizzo e' simile ad un comune oscilloscopio (vedi Figura 2). Sono disponibili comandi per modificare l'ampiezza e l'estensione temporale, CUT & PASTE, LOAD, SAVE.

6. Le Utility ENS

ENS e' una raccolta di programmi e librerie per l'elaborazione numerica dei segnali sonori (vedi figura 3). Filtri Numerici: FIR, IIR, Median, analisi LPC, adaptive LMS, Noise Reduction, Comb.

Trasformata veloce di Fourier.

Generatori di segnali: per sintesi di Fourier, inviluppi, funzioni distorcimenti, e rumori .

Rivbero digitale.

Visualizzatori di segnale: a 3 dimensioni, istogramma, spettro.

MIX e DEMIX di files sonori.

Phase Vocoder

Windowing

Convertitori di formato qualsiasi (floatsam/shortsam/ASCII/USER)

Possibilita' di multielaborazione e elaborazione multicanale. Tali programmi elaborano file in formato floatsam con o senza header.

E' disponibile un esempio di interfaccia con schede DSP (Texas TMS320C30) per la generazione di segnali mediante sintesi di Fourier.

Le librerie contengono funzioni di utilita' per l'elaborazione numerica dei segnali [3].

7. Il gestore di memoria virtuale V.M.S. (Virtual Memory System)

V.M.S. e' una libreria di funzioni scritte in Linguaggio C per la gestione virtuale della memoria a richiesta di pagina. E' configurabile in quanto e' possibile specificare la dimensione di una pagina (max 64Kb), il numero massimo di pagine in memoria, l'utilizzo eventuale delle aree di swap (memoria estesa (XMS), memoria espansa (EMS), memorie di massa), e il tipo di algoritmo di sostituzione desiderato. V.M.S. utilizza per l'accesso ad un blocco di memoria indirizzi

virtuali a 32 bit portando quindi la dimensione della memoria principale a 4Gb virtuali. V.M.S. e' stato utilizzato con CMUSIC ed esempi dettagliati di utilizzo si trovano in [7].

8. Il linguaggio CMUSIC per MS-DOS

L'attuale versione (2.1) di CMUSIC presenta le stesse caratteristiche offerte dalla versione originale per UNIX [2], con l'aggiunta dell'unita' di generazione per la sintesi granulare [4]. In particolare CMUSIC traduce uno spartito (un semplice file di testo) in un file di campioni sonori. Lo spartito contiene la definizione degli strumenti (combinazioni di unita' di generazione), generatori di funzioni e la lista delle note. Dato che il sistema operativo MS-DOS, limitando la gestione della memoria principale ad un massimo di 640Kb impedisce la scrittura di spartiti con un elevato numero di strumenti o con strumenti particolarmente complessi [7], si e' ricorso al gestore V.M.S. Ma dato che questi, come tutti i gestori di memoria virtuale, rallenta inevitabilmente le prestazioni del sistema sono state realizzate 3 versioni di CMUSIC, una per spartiti semplici, una per spartiti

con molti strumenti e una per strumenti particolarmente complessi. Nel primo caso non viene utilizzato V.M.S., nel secondo caso l'intera lista degli strumenti utilizzati viene posta da V.M.S. in memoria virtuale, e nel terzo caso vengono poste in memoria virtuale solo quelle unita' di generazione che richiedono numerosi buffers per la memorizzazione temporanea dei campioni elaborati.

9. Conclusioni

Il software presentato e' soggetto a sviluppi continui. La semplicita' e la flessibilita' di utilizzo lo rendono aperto a tutte le possibili soluzioni di miglioramento delle prestazioni. E' disponibile presso il LIM-DIST dell'Universita' di Genova un manuale dettagliato del sistema descritto [7].

BIBLIOGRAFIA

- [1] A.Camurri, G.Haus: "*Architettura e Ambienti operativi della stazione di lavoro intelligente*", IX Colloquio di Informatica Musicale - Genova, Novembre 1991.
- [2] Computer Audio Research Laboratory: "*CARL Startup Kit*" University of California, San Diego 1985.

[3] "*Programs for Digital Signal Processing*", Edited by the Digital Signal Processing Committee, IEEE Acoustics, Speech and Signal Processing Society, IEEE Press, 1979.

[4] D.L.Jones, T.L.Parks "*Generation and Combination of Grains for Music Synthesis*", Computer Music Journal Vol 12, N.2 summer 1988.

[5] "*Hdmusic: Interfaccia PC scheda PITDAD*", Rapporto Interno DIST - Laboratorio di informatica Musicale, Universita' di Genova.

[6] A.Camurri, F.Giuffrida, P. Podesta': "*DMTOOL: un Ambiente Software per l'elaborazione di campioni*", Atti VIII CIM Ottobre 1989 Cagliari.

[7] P.Fischetti: "*PC-MUSIC: Una stazione di lavoro per l'elaborazione di suono e musica*", Rapporto Interno DIST - Laboratorio di informatica Musicale, Universita' di Genova.

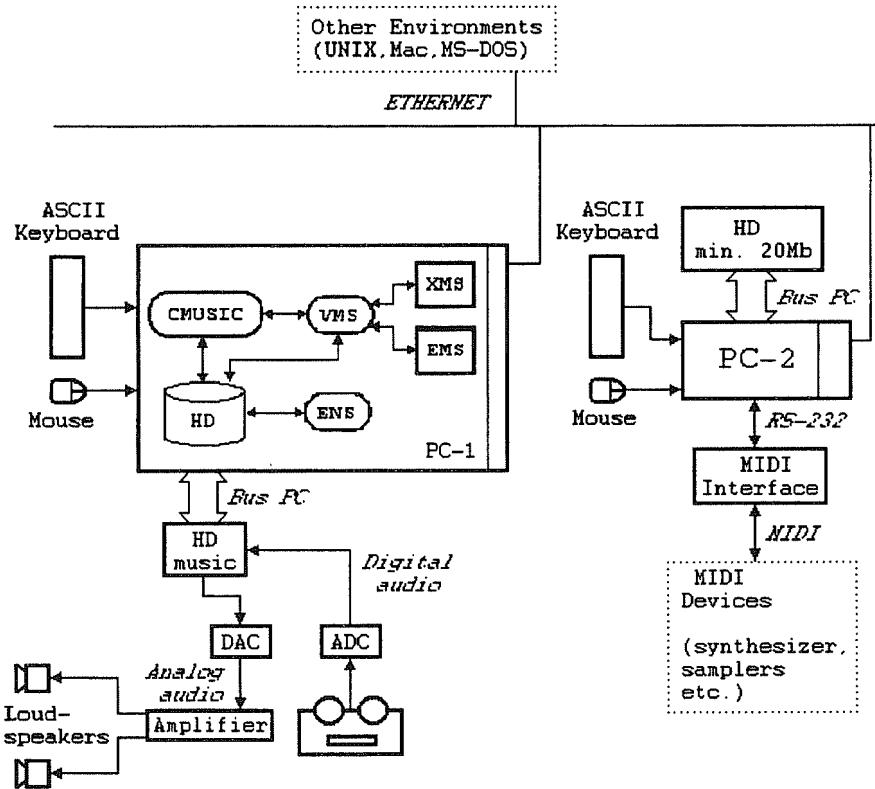


Figura 1 - Il sistema PC-MUSIC.

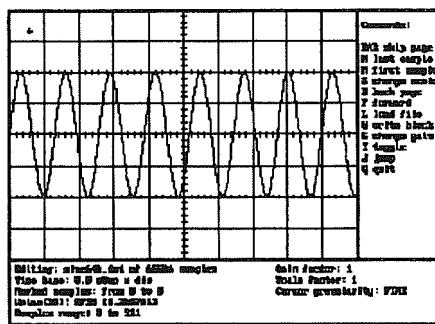


Figura 2 - Il programma GED.

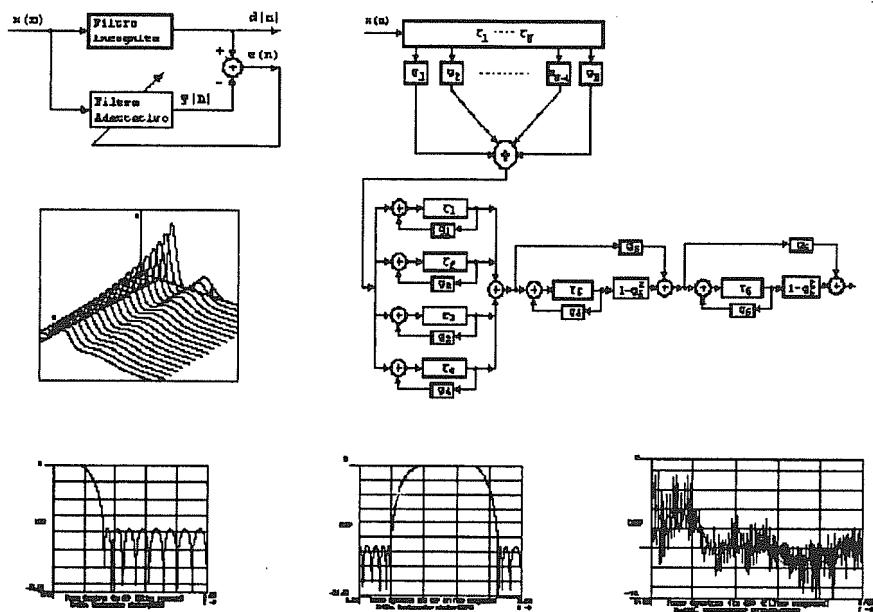


Figura 3 - Alcuni programmi contenuti nel modulo ENS

"STAZIONE DI LAVORO MUSICALE INTELLIGENTE": L'AMBIENTE INTEGRATO MACINTOSH-NEXT

Goffredo Haus, Isabella Pighi

Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39
I-20135 Milano (Italia)
fax +39 2 55006373
e-mail: music@imiucca.csi.unimi.it

Abstract

The Macintosh/NeXT Integrated environment of IMW is a set of both software & hardware modules hierarchically structured and communicating each other by protocols and standard formats.

The Symbolic-Structural Environment consists of six software modules for Macintosh: ScoreSegmenter, ModelSynth, ScoreSynth, Functional Performer, MusSer and Sequencer MIDI/DSP. The main developments occurred during last two years mainly affect ModelSynth, ScoreSynth and Functional Performer modules.

The Operating-Executing Environment consists of six software modules for either Macintosh or NeXT: Driver MIDI/DSP, LASy Workbench, Waver, TimbreLab, CASP (Cellular Automata Sound Processor) and VoiceLab. The main developments occurred during last two years mainly affect Driver MIDI/DSP, LASy, TimbreLab, CASP and VoiceLab modules.

In this paper we give an architectural overview of the current state of the project and a survey of all the software modules we have developed.

General architectural characteristics and high level specification of the modules are specified with more details in the MUSIC Tech. Report Series published by the Italian National Research Council.

This research has been supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

Architettura della SLMI.

La Stazione di Lavoro Musicale Intelligente (SLMI) è un ambiente integrato Macintosh-NeXT costituito da un insieme di moduli hardware e software organizzati in modo gerarchico e tra loro comunicanti mediante un insieme di protocolli e standard di formati [1]. Nei primi due diagrammi della pagina seguente sono riportati i collegamenti funzionali esistenti all'interno della SLMI a livello simbolico/strutturale e a livello esecutivo/operativo.

A parte i comandi MIDI, che attualmente costituiscono l'unico tipo di comunicazione fra i diversi ambienti h/w-s/w della SLMI (Macintosh e NeXT), i moduli comunicano mediante file il cui tipo di formato è specificato nel seguente elenco:

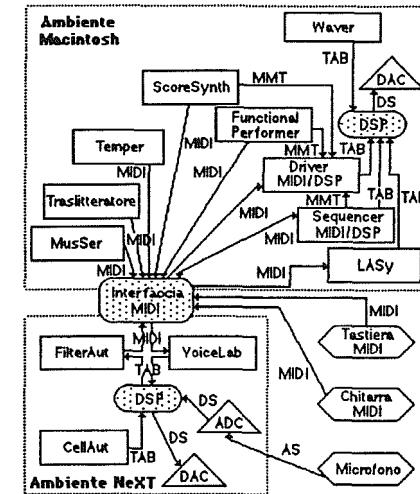
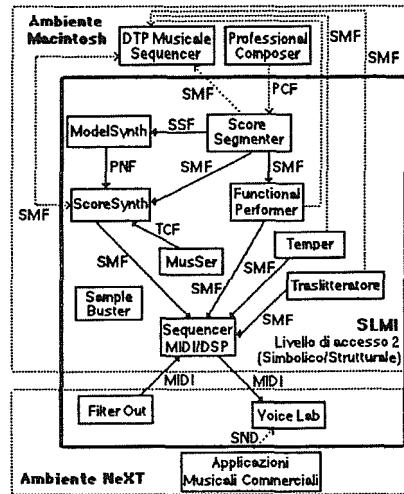
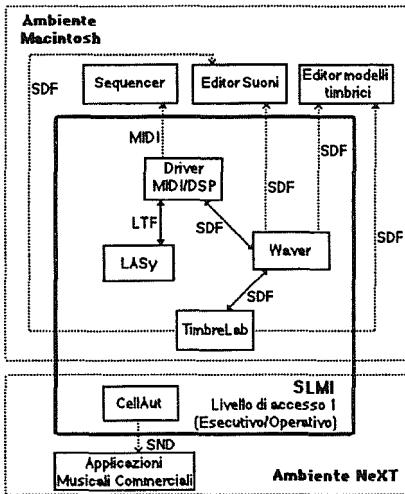
- SMI: Standard MIDI File formato 0 e 1;
- PNF: file utilizzati da ScoreSynth contenenti modelli di Reti di Petri;
- TCF: file di formato Tab Converter;
- PCF: file di formato Professional Composer;
- SSF: file prodotti da ScoreSegmenter durante la fase di analisi;
- SDF: file di formato Sound Designer;
- LTF: file prodotti da LASy Workbench;
- SND: file sonori di formato NeXT.

Il terzo diagramma illustra le funzionalità operanti in tempo reale della SLMI [2], i messaggi presenti nei canali di comunicazione sono:

- MIDI: messaggi MIDI;
- TAB: parametri di sintesi;
- DS/AS: segnale digitale/analogo;
- MMT: comunicazione virtuale via MIDI Management Tools.

I collegamenti illustrati sono in verità virtuali in quanto all'interno della SLMI alcuni moduli entrano in concorrenza specialmente per quanto riguarda l'utilizzo delle schede DSP; tali vincoli portano alle seguenti conclusioni:

- Waver, LASy e il Driver MIDI/DSP richiedono l'uso del medesimo DSP, sono quindi mutuamente esclusivi [3];
- ScoreSynth, Temper, MusSer, Traslitteratore, Functional Performer non possono essere utilizzati contemporaneamente sul medesimo calcolatore in quanto richiedono una continua interazione con l'utente e quindi un continuo utilizzo del display;
- il Sequencer MIDI/DSP può comunicare mediante MMT con tutte le applicazioni che possiedono tale tipo di interfaccia;
- il Driver MIDI/DSP può essere utilizzato concorrentemente al Sequencer MIDI/DSP in quanto il Sequencer MIDI/DSP opzionalmente disattiva il possesso della scheda;



1. Livello S/W 1:

Moduli a basso livello per le attività di analisi-sintesi del suono, la costruzione e l'elaborazione di campioni e modelli timbrici e la produzione di master audio numerici.

2. Livello S/W 2:

Moduli ad alto livello per le attività di composizione, analisi/sintesi di testi musicali e di strutture di testi musicali, performance musicale e audio visuale, orchestrazione, DTP musicale.

3. Funzionalità operanti in tempo reale della SLMI.

inoltre, possono essere eseguiti concorrentemente tante istanze di Driver MIDI/DSP, ciascuno con un microprogramma indipendente dagli altri, quante sono le schede DSP disponibili;

- il Driver MIDI/DSP comunica mediante MMT con tutte le applicazioni interfacciabili;

- FilterAut, CellAut, VoiceLab richiedono l'uso dello unico DSP disponibile attualmente in ambiente NeXT, ciò esclude l'esecuzione concorrente sul medesimo calcolatore;

- l'interfaccia MIDI mette in comunicazione le applicazioni agenti su diversi calcolatori, è possibile infatti una eventuale comunicazione fra i due ambienti controllando, ad esempio, il modulo VoiceLab mediante il Sequencer MIDI/DSP.

I moduli della SLMI

Interfaccia SLMI

Interfaccia SLMI è una interfaccia utente di tipo ipertestuale per la Stazione di Lavoro Musicale Intelligente. È caratterizzata da un alto grado di ipermedialità e di interazione e da una grafica iconica intuitiva. L'interfaccia consente di utilizzare i programmi commerciali di proprio interesse insieme con tutti i moduli della SLMI o con una parte di essi, definendo così opportuni ambienti operativi specialistici (per composizione, musicologia, editoria, produzione, etc.).

ScoreSynth 3.0

ScoreSynth 3.0 [4] consiste in un Editor/Esecutore/Debugger di Reti di Petri Musicali (RPM). Partendo dalla definizione di Reti di Petri PT (posti/transizioni) temporizzate dotate di capacità sui posti, di molteplicità sugli archi e morfismi di tipo raffinamento, alla definizione delle RPM si è giunti associando ai nodi di tipo posto il concetto di Oggetto Musicale e ai nodi di tipo transizione quello di Algoritmo di trasformazione degli Oggetti Musicali stessi. Un Oggetto Musicale è una sequenza di messaggi MIDI; un Algoritmo è una sequenza di Operatori Musicali applicabili sia ai parametri che specificano una nota: altezza, intensità, durata, timbro (canale MIDI); sia all'ordine in cui le note sono disposte all'interno di un Oggetto Musicale.

Score Segmenter

Obiettivo dell'applicazione è la segmentazione di brani musicali quale primo passo per una varietà di applicazioni musicologiche ed in particolare per la strumentazione automatica mediante elaboratore. Si tratta cioè della ricerca ed identificazione dei vari oggetti musicali e delle relazioni tra di essi di cui è composto il brano.

ModelSynth

E' in corso la realizzazione del modulo ModelSynth, il quale

permette di ricavare automaticamente un modello di Reti di Petri a partire dall'analisi effettuata da ScoreSegmenter. I modelli di Reti di Petri prodotti sono a loro volta utilizzati per eventuali analisi, eseguiti e/o manipolati dal modulo ScoreSynth. Possiamo quindi dire che ModelSynth rappresenta un anello di congiunzione tra l'analisi e la sintesi di partiture musicali.

MusSer

MusSer è un modulo per la generazione di stringhe numeriche/musicali di natura seriale su scale temperate a temperamento pari, nonché del reperimento di quelle serie che soddisfano particolari proprietà musicali.

Functional Performer

Il modulo Functional Performer [5] si configura come strumento per la performance funzionale in tempo reale di oggetti musicali.

Le strutture musicali costituenti le primitive su cui operare sono definite come frammenti melodici cioè sequenze di note musicali caratterizzate da altezza, durata e intensità. Ai frammenti melodici sono associati gli oggetti grafici che ne permettono la manipolazione da parte dell'utente attraverso gli operatori di trasformazione implementati. Gli operatori di trasformazione modificano i frammenti sele-

zionati fornendo il risultato della loro applicazione in tempo reale.

Il modulo accetta in input una serie di frammenti musicali codificati in forma di Standard MIDI File. A tal fine sono state utilizzati le routine ed il ciclo di lettura di SMF del modulo Libreria di routine di lettura/scrittura di Standard MIDI File 1.0.

Temper

Temper [6] è un'abbreviazione per TEssellating Music PERformer, ovvero esecutore di musica generata automaticamente a partire da animazioni grafiche di tipo tassellazioni. Deve intendersi per tassellazione una combinazione teoricamente infinita (ma in pratica limitata dalle dimensioni del disegno) di figure di uno o più tipi sul piano, in maniera tale che i loro contorni combacino perfettamente.

Mediante l'assegnazione di opportuni parametri, Temper permette la generazione e la produzione di sequenze musicali rigidamente determinate dalle tassellazioni che le generano, senza alcun ulteriore intervento umano; la corrispondenza è quindi tale da giustificare l'uso del termine "tassellante" per questo tipo di musica.

Traslitteratore

L'applicazione consente di realizzare in modo automatico,

secondo alcune regole, la trasformazione di un testo letterario in uno spartito musicale.

Traslitteratore consente altresì di ascoltare i brani ottenuti dalla traslitterazione, sia attraverso i suoni prodotti con il sintetizzatore interno al Macintosh oppure mediante periferiche MIDI. Tale traduttore è in grado di tradurre anche i simboli di accelerando e ritardando che si possono inserire nel testo, ottenendo così di poter rimuovere, almeno in parte, sensazioni di appiattimento espressivo.

Sequencer MIDI/DSP

Il Sequencer MIDI/DSP è un MIDI Sequencer costruito per il controllo di una scheda DSP, che viene vista dall'utente come un sintetizzatore a cui è possibile inviare sequenze di dati MIDI come a un qualsiasi dispositivo MIDI esterno.

L'applicazione consente di effettuare registrazione, editing e riproduzione di sequenze MIDI distribuite su molteplici tracce (max 32). Queste possono essere singolarmente assegnate per pilotare l'esecuzione di dispositivi MIDI esterni e/o la scheda DSP interna; è anche possibile effettuare operazioni di mixing MIDI sulle tracce. E' possibile inoltre richiamare il modulo *Waver* e impostare i differenti

modelli timbrici delegati al controllo della scheda DSP.

Driver MIDI/DSP

Il programma DMD è un driver MIDI costruito per il controllo di una scheda DSP (Sound Accelerator, Audio-media).

Si può considerare questa applicazione come un naturale sviluppo del Sequencer MIDI/DSP; sono state incrementate le capacità di comunicazione a livello MIDI in tempo reale e sono state rese disponibili più potenti possibilità di controllo timbrico.

Ma la caratteristica principale è senza dubbio rappresentata dalla possibilità di inviare messaggi MIDI per il controllo della scheda DSP da parte di una qualunque applicazione compatibile con il MIDI Manager.

Waver

L'applicazione Waver è un 'banco di lavoro' per la sperimentazione di tecniche di sintesi; attualmente sono implementate le seguenti: sintesi mediante funzioni di due variabili, waveshaping, trasformata wavelet.

LASy Workbench

Il LASy Workbench è un'applicazione sviluppata per la realizzazione dell'algoritmo di sintesi LASy (Linear Automata Synthesis) su Macintosh dotato di una scheda

DSP (Sound Accelerator o Audiimedia).

Il LW risponde ad una duplice funzione: si tratta di uno strumento di esecuzione musicale LASy controllabile in tempo reale con comandi MIDI o con periferiche di ingresso (tastiera ASCII e mouse); inoltre costituisce un ambiente di ricerca e di sperimentazione della sintesi LASy.

TimbreLab

L'applicazione consiste in un sistema ad architettura aperta per la programmazione timbrica con interfaccia iconica. I modelli timbrici possono essere organizzati gerarchicamente; infatti, un modello può sempre essere considerato come un sotto-modello rappresentato da un nodo in un modello più complesso.

TimbreLab permette la realizzazione delle diverse tecniche di sintesi disponibili in letteratura.

VoiceLab

VoiceLab è il risultato di un'indagine nel campo della sintesi della voce in generale e del parlato in particolare, con controllo dell'andamento intonativo del parlato sintetico. VoiceLab è eseguibile in ambiente NeXT ed è costituito da una parte di codice che viene eseguita sul processore principale e da un modulo di sintesi del segnale vocale scritto

in microcodice del DSP Motorola 56001.

Cellular Automata Sound Processor

CellAut (eseguibile in ambiente NeXT) sintetizza un segnale utilizzando l'algoritmo LASy a partire da una forma d'onda e una funzione di transizione scelte dall'utente. Quest'ultimo ha un controllo duplice sull'evoluzione della sintesi: i parametri della funzione di transizione sono modificabili tramite oggetti grafici e l'altezza del suono prodotto è controllata dall'altezza di un segnale analogico rilevato da un microfono.

Parallelamente a CellAut, è stata sviluppata una seconda applicazione, FilterAut, che consente di filtrare in tempo reale un segnale analogico e contemporaneamente di variare i parametri di filtraggio. Inoltre FilterAut calcola l'altezza del segnale e invia la nota corrispondente come una sequenza di comandi sul canale MIDI collegato all'elaboratore.

La libreria di microprogrammi di sintesi

Il presente modulo consiste in un insieme di routine per la sintesi del suono utilizzato da quattro software della stazione di lavoro: Driver MIDI/DSP, Sequencer MIDI/DSP, Waver, LASy Workbench.

Tali routine sono rese disponibili sotto forma di

risorsa nel file "DSP Code Resources".

Standard MIDI files Library

Si tratta di una libreria sviluppata per elaboratori Macintosh e fornisce un insieme di routine per la gestione dei processi di lettura e scrittura di Standard MIDI File di formato '0' e '1' seguendo la sintassi definita nel documento Standard MIDI File V. 1.0 pubblicato dalla International MIDI Association.

SampleBuster

L'applicazione consiste nella gestione ragionata e guidata di una raccolta di campioni audio provenienti da diverse biblioteche e memorizzati con diversi formati fisici, utilizzabili in ambienti di produzione musicale professionali e non, con alcune utility per la gestione veloce delle apparecchiature esterne, collegate al sistema.

Ovviamente è possibile utilizzare campioni provenienti da qualsiasi biblioteca, essendo i formati scelti di carattere assolutamente generale.

Bibliografia

[1] AA.VV. (A. Stiglitz Ed.): "*Specifiche funzionali di alto livello dei moduli della Stazione di Lavoro Musicale Intelligente - Unità Operativa della Università degli Studi di*

Milano", Tech. Report 7/53, CNR-PFI2, 1991.

[2] I. Pighi & AA.VV.: "*Integrazione dell'architettura e delle specifiche funzionali di alto livello dei moduli della Stazione di Lavoro Musicale Intelligente - Unità Operativa dell'Università degli Studi di Milano*", Tech. Report 7/117, CNR-PFI2, 1993.

[3] A. Ballista, E. Casali, J. Chareyron, G. Haus: "*A MIDI/DSP Sound Processing Environment for a Computer Music Workstation*", Computer Music Journal, Vol.16, N.3, pp.57-72, MIT Press, 1992.

[4] G. Haus, A. Sametti: "*SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra*", in "*Readings in Computer Generated Music*", D. Baggio Ed., pp.53-78, IEEE Computer Society Press, 1992.

[5] G. Haus, A. Stiglitz: "*The Functional Performer System*", Interface, Vol.23, N.1, pp.53-75, Swets & Zeitlinger B.V., 1993.

[6] G. Haus, P. Morini: "*TEMPER: a System for Music Synthesis from Animated Tessellations*", Leonardo, Vol.25, N.3/4, pp.355-360, Pergamon Journals, 1992.

L'AMBIENTE PER L'ANALISI/RE-SINTESI DI PARTITURE DELLA "STAZIONE DI LAVORO MUSICALE INTELLIGENTE"

Goffredo Haus, Alberto Sametti

Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39
I-20135 Milano (Italia)
fax +39 2 55006373
e-mail: music@imiucca.csi.unimi.it

Abstract

In this work we describe the analysis/re-synthesis environment of the "Intelligent Music Workstation". It consists of three software modules:

- a) *ScoreSegmenter*, which is able to decompose a score into a set of basic music objects and a number of transformation relationships among various occurrences of basic music objects within the score; it tries to recognize the main theme of the piece, if any exists; then it finds any instance of the main theme considering also any possible transformations of it; both entire and partial instances are considered; this automatic process is controlled through many configuration parameters the user can set;
- b) *ModelSynth*, which synthesizes a Petri Net model representing the generative structures of the original score; it starts from informations extracted by ScoreSegmenter such as the number of voices within the score, the presence of loops, the application of music transformation (transposition, retrogradation, mirror inversion, ...), etc.;
- c) *ScoreSynth*, which executes that Petri Net model either resynthesizing the original score or synthesizing new scores, depending on eventual editing of the formal model.

This research has been supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

Introduzione

L'ambiente per l'analisi/re-sintesi di partiture costituisce il livello più astratto (strutturale/simbolico) della "Stazione di Lavoro Musicale Intelligente" [1].

Esso fornisce strumenti interattivi per la scomposizione automatica di partiture, per la sintesi di modelli generativi basati sui risultati della scomposizione e per l'esecuzione di tali modelli.

E' un ambiente mirato allo incremento della produttività individuale tanto del musicologo quanto del compositore.

Consente infatti di individuare strutture nascoste nelle partiture e di utilizzarle o per dare una rappresentazione più astratta e strutturata delle partiture stesse o per generare varietà di partiture con caratteristiche strutturali più o meno (a piacere del musicista) affini a quelle di una partitura data.

L'ambiente è costituito da tre moduli s/w: *ScoreSegmenter* (per la scomposizione), *ModelSynth* (per la sintesi dei modelli generativi), *ScoreSynth* (per la esecuzione dei modelli ovvero per la sintesi di nuove partiture). E' bene sottolineare che l'esecuzione con *ScoreSynth* di un modello generato da *ModelSynth*, senza che il musicista vi apporti modifica alcuna, avrà come effetto la generazione della partitura di partenza che era stata scomposta con *ScoreSegmenter*.

Architettura software

In questo capitolo analizzeremo quella che è l'architettura software più generale, cioè il contesto in cui si inserisce il modulo centrale, *ModelSynth*. Come si vede in Fig. 1, direttamente correlati con esso esistono gli altri due moduli *ScoreSegmenter* [2] [3] e *ScoreSynth* [4]. Illustriamo brevemente le peculiarità dei tre moduli.

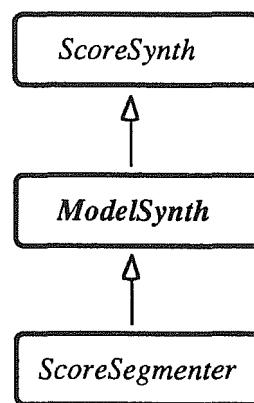


Fig. 1: architettura moduli s/w.

ScoreSegmenter si pone lo obiettivo della segmentazione di brani musicali quale primo passo per una futura loro strumentazione automatica proposta dall'elaboratore. Si tratta cioè della ricerca dei vari oggetti musicali con cui è composto il brano, dove per oggetti intendiamo quei frammenti musicali che l'autore ha espresso e che ha ripreso e trasformato, secondo i vari canoni musicali, in relazione al periodo storico e alla

forma compositiva. E' uno strumento informatico prototipale che consente di accostarsi ai testi musicali non come pura sequenza di note, bensì vedendo i testi stessi come un edificio espressivo fatto da alcuni elementi di base strutturati funzionalmente fra loro in vario modo. ScoreSegmenter è quindi essenzialmente uno strumento di analisi.

ScoreSynth al contrario è uno strumento di sintesi. Esso rende disponibile un ambiente integrato per la creazione, debugging ed esecuzione di modelli di partiture

musicali basati sul formalismo delle reti di Petri. L'obiettivo primario del modulo ScoreSynth è quello di mettere a disposizione del 'musicista' uno strumento molto potente ed efficace che permette di considerare l'attività compositiva da un punto di vista architettonicale: il 'musicista' manipola sequenze, funzioni di trasformazione e strutture.

Infine, ModelSynth permette di tradurre, in termini di modelli di reti di Petri eseguibili da ScoreSynth, i risultati dell'analisi operata dal modulo ScoreSegmenter.

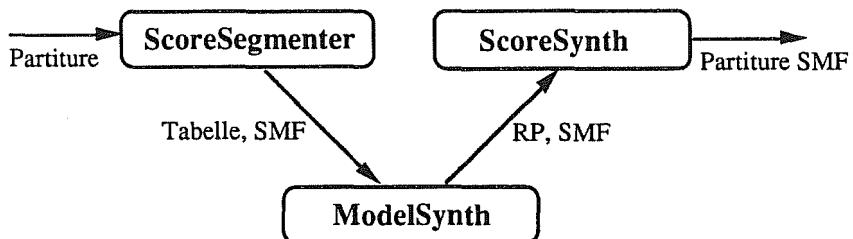


Fig. 2: flussi di Input/Output.

Specifiche funzionali e input/output

Considerando i tre moduli da un punto di vista funzionale vediamo ora quali trasformazioni definiscono e cosa costituisce il dominio e il codominio per ciascuni di essi.

ScoreSegmenter

ScoreSegmenter è in grado di leggere una partitura espressa in

notazione musicale tradizionale e generare un file di supporto (detto 'file di lavoro'), corrispondente al file in ingresso, tramite cui operare la propria analisi, i vari temi riconosciuti in formato SMF (Standard MIDI File 1.0) e, più importante, una tabella testuale contenente i risultati delle ricerche.

Quest'ultima tabella, per ogni tema o frammento trovato, riporta le seguenti informazioni: la *voce*, la *nota iniziale* e la *nota*

finale all'interno della partitura, l'*operatore* eventualmente applicato e la *prima nota* riconosciuta del tema. D'ora in poi faremo riferimento ad ogni riga della tabella con il termine 'atomo'.

Affinchè le tabelle dello ScoreSegmenter siano significative si devono fissare i parametri di impostazione della segmentazione in modo da effettuare ricerche esclusive su frammenti e temi. In modo da evitare che una sequenza di note, in una certa posizione nella partitura, appartengano a più temi o frammenti.

ModelSynth

ModelSynth legge i frammenti riconosciuti e memorizzati in formato SMF e la tabella testuale prodotta dall'analisi dello ScoreSegmenter; li analizza iterativamente al fine di individuare strutture descrivibili mediante reti di Petri; infine, sintetizza un modello di reti di Petri generativo gerarchico, eseguibile da ScoreSynth, avente i frammenti in formato SMF associati ad opportuni nodi del modello.

ScoreSynth

Come detto ScoreSynth permette l'*editing*, il *debugging* e la esecuzione di modelli di reti di Petri orientati alla sintesi di partiture musicali MIDI. Per una dettagliata descrizione delle fun-

zionalità si rimanda all'articolo citato in bibliografia.

Criteri di analisi del modulo ScoreSegmenter.

Un primo aspetto fondamentale del modulo consiste nella realizzazione degli algoritmi per la ricerca delle occorrenze degli oggetti musicali, o di loro sottoparti, all'interno del brano (oggetti che possono essere individuati dall'elaboratore stesso o forniti dall'utente). Come noto, a questo proposito sono due gli elementi da considerare: gli attributi della singola nota e la trasformazione musicale che la nota stessa ha subito assieme alle note che la precedono e/o seguono. Gli attributi sono la durata, l'accento, il nome della nota, l'altezza in semitonni e l'andamento dell'intervallo. Per quanto riguarda le trasformazioni, nel presente lavoro sono state considerate quelle applicate agli attributi di posizione, o grado, sia nella scala tonale che nella scala diatonica. Tali trasformazioni sono state realizzate attraverso tre tipi diversi di operatori algebrici e loro combinazioni, precisamente si tratta dell'*operatore* di *trasposizione*, di *inversione speculare* e di *retrogradazione*. Tali operatori applicati, come detto, al nome delle note, realizzano le corrispondenti trasformazioni musicali tonali, applicati all'altezza in semitonni, le trasformazioni reali. Per fornire

buona duttilità a questo strumento di analisi, si è reso possibile intervenire, in mondo interattivo, per variare notevolmente lo 'stile' con cui le ricerche sono condotte, scegliendo quali attributi considerare, quali trasformazioni e quale tasso di variabilità applicare alle analisi tonali.

Un secondo aspetto è la segmentazione vera e propria. Per semplicità ci siamo basati sulla forma musicale della fuga, generalizzandola poi per le altre forme, anche se sarebbe opportuno ampliare l'algoritmo al fine di individuare, con maggior precisione e specificità, il tema vero e proprio del brano nelle altre forme. Per la individuazione degli oggetti ci si è basati non solo sulle loro ripetizioni, che sono evidentemente un aspetto necessario ma non sufficiente ai fini musicali. Un tema ha infatti anche un impianto tonale (ad eccezione della musica moderna) e metrico dai quali non è possibile prescindere. Per tener conto anche di questi aspetti vengono formulate delle richieste per le note degli ipotetici temi, in relazione all'appartenenza e affermazione della tonalità a seconda tel tipo di oggetto (tetico, acefalo e anacrusico) e in relazione alla esaustività e compiutezza del pensiero musicale, quale effetto della metrica. Anche per questa fase di individuazione dei temi si è predisposta una serie di parametri da specificare interattivamente sia circa gli elementi da valutare

nella definizione del tema (tonalità, metrica, lunghezza del fraseggio, numero minimo di ripetizioni, ecc.), sia circa la ripetizione stessa degli oggetti.

Criteri di analisi/re-sintesi del modulo ModelSynth.

In sostanza ciò che ModelSynth compie è una operazione inversa rispetto a quella dello ScoreSegmenter. Infatti cerca di ricostituire la struttura del brano musicale analizzato e atomizzato. Chiaramente questa operazione per non essere un passaggio ridondante deve conferire alla struttura finale del brano alcune caratteristiche significative. Queste caratteristiche si possono riassumere in una sola: rappresentare con modelli flessibili il contenuto informativo del brano, dove per 'informativo' intendiamo le relazioni tra gli atomi costituenti e le loro trasformazioni nel corso dello svolgimento. Ciò che viene realizzato è quindi l'analisi di una analisi, allo scopo di, da una parte, fornire una rappresentazione alternativa, dall'altra estrarre e codificare l'informazione per la resintesi.

La prima analisi compiuta da ModelSynth è quella di operare il parsing della tabella prodotta dallo ScoreSegmenter, generata a seguito di una richiesta di segmentazione di un brano, utilizzando come supporto il file di lavoro relativo. Questa fase permette la rappresentazione in

un formato intermedio dei risultati contenuti nelle tabelle. Ovvero salva tutti i temi riconosciuti assegnando un codice di riferimento, suddivide tutti gli atomi per voce, riordina le loro occorrenze secondo una chiave temporale e recupera dal file di lavoro tutte quelle parti (sequenze di note e pause), che possiamo chiamare ‘scarti’, giudicati non significativi dallo algoritmo di segmentazione, ma che costituiscono la colla tra i vari atomi.

Secondariamente vengono considerati gli operatori applicati agli atomi. Poiché gli operatori riconosciuti dallo ScoreSegmenter sono un sottoinsieme stretto di quelli disponibili in ScoreSynth, questa operazione è immediata.

A questo punto ModelSynth ha a disposizione tutte le informazioni necessarie per operare la propria analisi e quindi la costruzione automatica di un modello di Reti di Petri, in formato ScoreSynth, corrispondente al brano di partenza e che metta in evidenza, se presenti, i costrutti relazionali e le funzioni di trasformazioni utilizzate. Fondamentale in questa fase è la possibilità di sfruttare i meccanismi di chiamate parametriche a reti gerarchiche messa a disposizione da ScoreSynth. Questi realizzano la separazione tra struttura del brano e temi. La struttura viene identificata dalle relazioni tra i temi, le loro ripetizioni e le loro trasformazioni, e viene quindi

codificata in un modello. I temi svolgono solamente il ruolo dei dati.

L'analisi procede considerando singolarmente le voci. Le strutture che si cerca di riconoscere sono i *loop semplici*, i *loop con selezione* e la *ripetizione di pattern* specifici.

Con loop semplici intendiamo le successioni di un tema o di una sua trasformazione. Ad esempio, considerando il tema A e un operatore T, una successione come

$$T(A) - T(A) - T(A) - T(A)$$

può essere rappresentata da una rete macro che realizzi un loop alla quale vengono passati come parametri il tema, l'operatore e il numero delle ripetizioni.

Un loop con selezione invece si ha se varia l'operatore applicato ad un medesimo tema. Ad esempio, nella successione:

$$T(A) - R(A) - I(A)$$

Questa tipologia di costrutto utilizza sempre come base una rete di tipo loop, con la differenza che l'oggetto del looping non è un singolo elemento, ma una sottorete macro alla quale vengono passati come parametri gli operatori da applicare ad ogni ciclo.

Oltre alla risoluzione dei loop vanno riconosciuti anche i pattern. E i pattern possono essere ricercati a diversi livelli. Al livello degli operatori; ad

esempio in successioni relative ad uno stesso tema come:

$$\begin{array}{l} T(A) - R(A) - T(A) - \\ R(A) - T(A) - R(A) \end{array}$$

Al livello dei temi; come in

$$\begin{array}{l} T(A) - R(B) - I(C) - \\ T(A) - R(B) - I(C) \end{array}$$

O infine a livello di intere reti. Le fasi più significative della analisi del ModelSynth si possono quindi riassumere mediante i seguenti passi:

- i) creazione di una rappresentazione intermedia;
- ii) riconoscimento di pattern sugli operatori;
- iii) riconoscimento di loop semplici o con selezione;
- iv) riconoscimento di pattern sui temi;
- v) riconoscimento di pattern su intere reti;
- vi) torna a v).

Il processo di analisi continua solamente se il passo v) è in grado di operare almeno un riconoscimento.

Nelle reti di Petri raffigurate nelle figure seguenti viene mostrato quale potrebbe essere il risultato, almeno per quanto riguarda gli aspetti più generali, dell'analisi operata da ModelSynth su un brano di nome "PieceX". In particolare la figura 3 rappresenta la rete di più alto livello del modello; la figura 4 la sottorete relativa alle parti in gioco (in questo esempio, tre); e, la figura 5, la rete che 'raccoglie'

tutti i temi completi riconosciuti da ScoreSegmenter. La figura 6, invece, mostra come nel modello può essere rappresentata la applicazione di un operatore ad un tema: l'operatore nella riga sottostante indica che la trasformazione da esso operata fa riferimento al tema "Theme2". Questa semplice rete potrebbe essere invocata più volte in un modello. Pertanto verrebbe

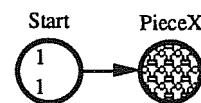


Fig. 3: la rete generata di livello più alto.

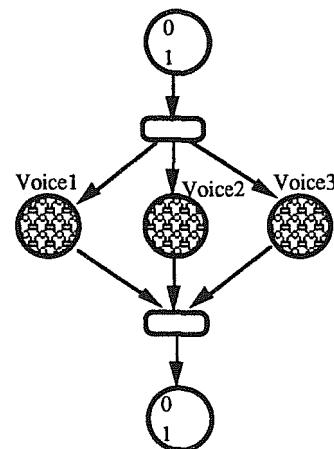


Fig. 4: la rete "PieceX".

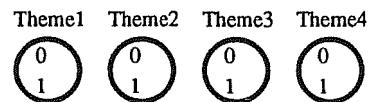
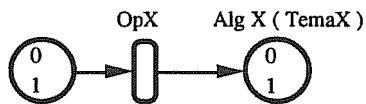


Fig. 5: la rete con tutti i temi riconosciuti.



$\text{OpX} = \text{P: } 1, \$, [\text{Tema2}, 1], ? + 7$

Fig. 6: un esempio di applicazione di un operatore di trasposizione ad un tema.

Futuri sviluppi

L'ambiente per l'analisi/re-sintesi di partiture qui brevemente descritto è da considerare un prototipo per la ricerca sperimentale che avrà tante più possibilità di accrescere la finezza delle sue fasi di analisi quanta più sistematica sperimentazione potremo attuare, con le seguenti finalità primarie:

- I) generalizzazione della capacità di segmentazione dello ScoreSegmenter alle più svariate forme musicali; attualmente, per quanto sia in grado di scomporre qualsivoglia partitura, ha maggiore capacità, ovvero produce meno scarti, con brani in forma di fuga o di sonata;
- II) estensione delle capacità di analisi/re-sintesi del ModelSynth per l'individuazione di strutture sempre più profonde; in particolare, si può potenziare ulteriormente l'individuazione di strutture affini estendendo l'attuale nozione di rete macro in modo tale che una rete macro possa essere considerata come un parametro di un'altra rete macro; questa possibilità richiede

una coerente estensione del modulo ScoreSynth.

In termini ancor più generali, possiamo pensare ad un approccio, quale quello qui seguito nel caso delle partiture musicali, applicato al caso dei processi multimediali, dove si possano scomporre, organizzare in modelli generativi e sintetizzare processi costituiti da suoni, immagini e testi.

Bibliografia

- [1] I. Pighi e AA.VV.: "Integrazione dell'architettura e delle specifiche funzionali di alto livello dei moduli della Stazione di Lavoro Musicale Intelligente - Unità Operativa dell'Università degli Studi di Milano", Tech. Report M/42, CNR-PFI2 MUSIC Series, 1993.
- [2] F. Lonati: *Guida operativa del modulo "ScoreSegmenter"*, Tech. Report M/18, CNR-PFI2 MUSIC Series, 1991.
- [3] F. Lonati: *Note tecniche del modulo "ScoreSegmenter"*, Tech. Report M/19, CNR-PFI2 MUSIC Series, 1991.
- [4] G. Haus, A. Sametti: "SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra", in "Readings in Computer Generated Music", D. Baggio Ed., pp.53-78, IEEE Computer Society Press, 1992.

Real Time Processing and Performance using WinProcne/HARP

Claudio Massucco, Marco Mercurio and Giuliano Palmieri
DIST - University of Genova
Laboratorio di Informatica Musicale
Via Opera Pia 11/A -16145 Genova

This paper describes a composition project based on the utilization of the *WinProcne-HARP* system, a tool able to represent and process in real time music knowledge. The goal of this project is twofold: from one hand it aims at putting to work the new version of the system in a real compositional framework, from the other hand it follows the artistic goals of the composer. The system has been used in a concert at Palazzo Ducale (Genova), and in Villa Arconati (Milano). It will be used in multimedia theatrical event for the joint management of both humanoid animations and music.

WinProcne/HARP in brief

WinProcne/HARP is a software environment that allows the creation, updating and querying of knowledge bases using a high-level graphical environment implemented under Microsoft Windows 3.1 and available on PC 386/486 [3,4]. The system is based on a hybrid formalism, integrating an object-oriented concurrent environment dealing with sound signals (samples, MIDI messages) and

compositional algorithms, with a symbolic level based on multiple inheritance semantic networks language derived from KL_ONE [2], extended with temporal primitives. The system allows the definition of a formal, symbolic data base integrated with a set of *analogical experts*, a kind of "actor" system [1]. They can be activated by a special actor (the *Simulative Engine*), guided by inferences on the network. The analogical and symbolic levels are connected by an interface that defines the links between analogical experts, their activation order, and manages the interactions between the two levels. In other words, it is possible, once defined the symbolic data base that contains the information on the knowledge of the problem domain, to 'ground' this knowledge onto a set of analogical experts able to execute different kinds of transformation, and to activate them following a *scheduling* algorithm that uses the information of the symbolic level and updates the context according to the events occurring in real-time.

The environment

WinProcne has been used in this project to control three Proteus synthesizers and a workstation *MARS* (developed by *Iris Bontempi Farfisa*) through MIDI interface; for each instrument notes and MIDI controls are generated; some controls, for example "main volume", are used with the "obvious" meaning, while others are used to control the synthesis process, both in the envelope and during the execution of a single note. The controls associated to the algorithms have been planned for producing timbral evolution.

One of the main problems met by the composer to produce music by means of this architecture has been the control of a high number of parameters: for every instrument, four keyboards (one for each MIDI channel) and forty cursors (ten controls for each channel), should be controlled at the same time. WinProcne/HARP allows the real-time control of all these parameters.

The sonic *fascia*

The musical aim of the composer is to build a musical situation - the *fascia* - where a continuum of situations are combined according to a logic or metaphors, like those of the movement of particles in force fields. The concept of *fascia* can be classified according to different features: at the extremes

there are notes with long durations (even in the range of one minute) and a small amount of pitch "verticality". Variations are generated by controlling the synthesis algorithm and, at the opposite side, short and overlapping notes, with less work on the timbre.

The continuous evolution between the two extremes and inside the different musical moments are controlled by WinProcne/HARP. The most cryptical phase of the work has been the formalization of the logic that guides the different moments and their evolution: the composer uses, for the generation of new notes and the control of the evolving notes, rules that he himself is not able to describe clearly. The knowledge base describing this logic, together with the experts that generate the corresponding controls, have been built during long test sessions, in which the composer described his needs that were formalized through trials and errors. In this way we developed a compositional model.

The final result is an environment in which the composer can manipulate, interactively during a performance, some kind of "knobs", i.e. active cursors that work on bidimensional force fields (see figure 2). The term "active" means that the cursor, once placed, does not stay in the chosen position but move in a force field following a given movement law (e.g., toward a

minimum, the darken area in the figure). The cursors are used to control high level parameters (musical moments, note density, quantity of timbral controls); if the cursors move freely there are some predefined behaviors (for example, the cursor that controls the density of notes and the quantity of distortion has been set to swing, in some minutes, between high density/few controls and low density/a lot of controls). Anyway, in any moment the performer can manually move the cursor, thus either accelerating or delaying the evolution of the system or completely changing it. According to the state of the active cursors and the state of the performance (quantity and pitch of the active notes on each channel, volumes, used timbers), intermediate level requests (like "raise the volume on a given channel", "make more percussive the envelope for the next notes", "modify the timber on a channel") are generated. These requests correspond to new assertions in the symbolic subsystem. The management of the MIDI controls is done by the analogical experts: a timbre modification, for example, requires an expert generating a sequence of MIDI messages that modify the value of one or more controls in a continuous way.

Conclusions

WinProcne/HARP demonstrated to be a flexible tool for music

modeling and real-time performance. This experiment has produced significant feedback for identifying some weak points of our system model. A first public performance was held on April 1993 at Palazzo Ducale in Genova, and in a concert at Villa Arconati on June in Milano. *WinProcne* will be used for a theatrical performance on the subject "Frankenstein e Pinocchio", directed by Mario Jorio, at the theater "Altrove" in Genova (early 1994).

Bibliography

- [1] G.Agha, C.Hewitt, *Actors: A Conceptual Foundation for Concurrent Object-Oriented Programming*, in *Research Directions in Object-Oriented Programming*, B.Schrivener and P.Wegner (Eds.), Cambridge, MIT Press, 1987.
- [2] R.J.Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, 9, pp.171-216, 1985.
- [3] A.Camurri, C.Canepa, M. Frixione, and R. Zaccaria, "HARP: A framework and a system for intelligent composer's assistance", *IEEE COMPUTER*, Vol.24, No.7, pp.64-67, July 1991.
- [4] A.Camurri, C.Canepa, M.Frixione, C.Innocenti, C.Massucco, R.Zaccaria, "HARP: un ambiente ad alto livello per l'ausilio alla composizione", Atti *IX Colloquio di Informatica Musicale*, Genova, 1991.
- [5] F.Courtot, "CARLA: Knowledge Acquisition and Induction for Computer Assisted Composition", *Interface*, Vol.21, pp.191-217, Swets&Zeitlinger.

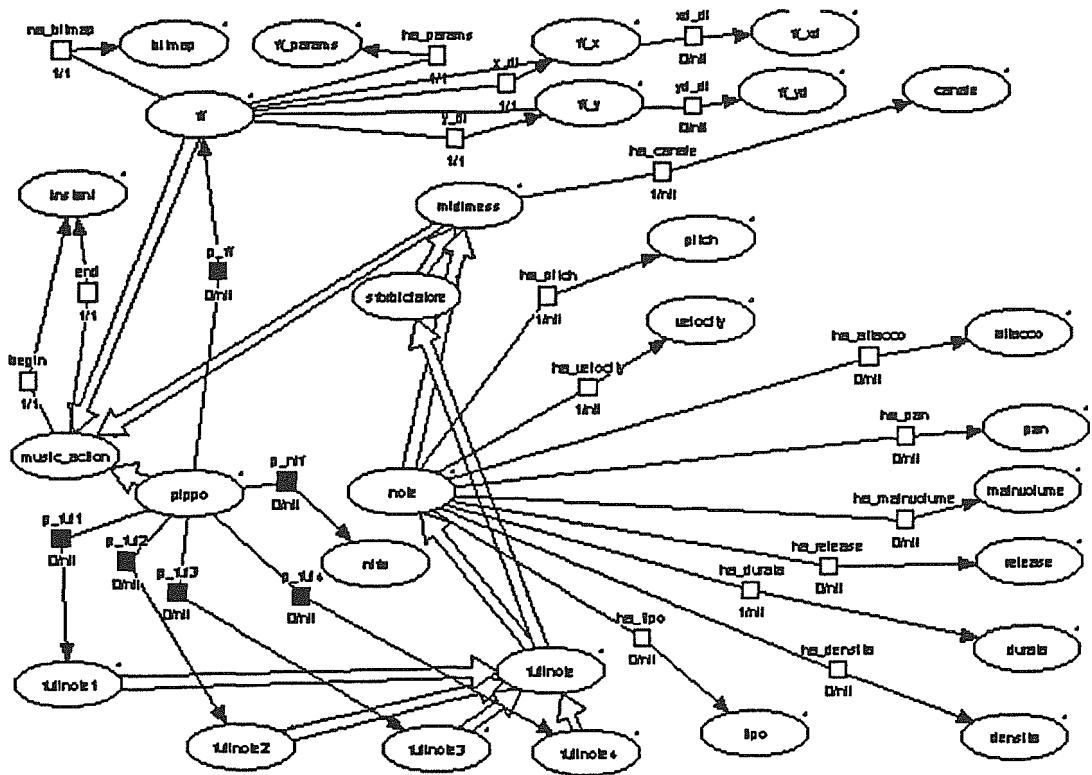


Figure 1: a fragment of the knowledge base.

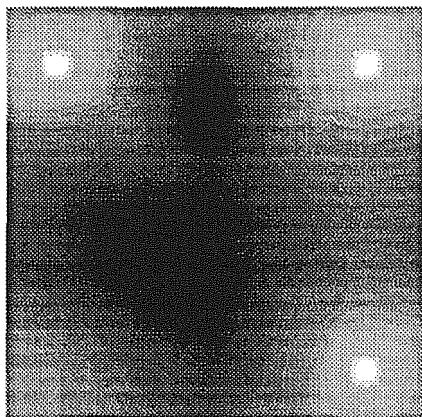


Figure 2: A force field for the production of a *fascia* object

Sezione 6b

MARS

APPLI20: A DEVELOPMENT TOOL FOR BUILDING MARS APPLICATIONS WITH AN EASY TO USE GRAPHICAL INTERFACE

P. Andrenacci, F. Armani, A. Prestigiacomo, C. Rosati
IRIS s.r.l.

Parco La Selva, 151
03018 Paliano (FR), Italy
fax: +39 (775) 533343 – tel: +39 (775) 533441
E-mail: mc2842@mclink.it

MARS is an integrated environment in which a graphical user interface, a realtime operating system and two general purpose digital signal processors are linked together to create a general sound processing system that is interactive and operates in realtime [1, 4].

EDIT20 provides a multi-purpose interactive graphical approach to define any kind of audio object, with immediate sound feedback [1, 4]. However, an application programmer may want to extend the existing MARS development environment on the host with his own programs [2].

APPLI20 is a tool to support MARS application programming on the host computer. It offers a programming approach to the MARS system through a set of libraries and interfaces that facilitate the development of MARS applications.

Why APPLI20?

The high level approach provided by EDIT20 [4] does not satisfy all MARS's possible users. Application programmers have more particular needs than those which EDIT20 can satisfy. The MARS environment on the host may require extension by new programs developed by IRIS and by the users themselves.

A MARS application program should satisfy some general requirements:

- to be an autonomous program;
- its user interface should be easily and quickly produced;
- to program and control the sound generation board or to create libraries of sounds and MIDI environments, without using EDIT20;

- to exchange data and cooperate with EDIT20 or other applications;
- to optimize the use of MARS resources, e.g. X20 microprograms and memories, when EDIT20 is not able to do it;
- to exploit the combined performance and capabilities of MARS and of the host computer to cover any aspect that EDIT20 does not or cannot solve (for example, to realize a realtime 2048 points FFT analysis with a graphical data presentation);
- to be coded by means of high-level standard programming languages;
- to reuse code previously developed and tested;
- to have a uniform style in coding and presentation.

In order to satisfy all the above requirements, IRIS offers to the application programmer APPLI20, a set of libraries with a well defined Application Programmer Interface, available in the C Language.

In APPLI20 the programmer finds tools to fully program and control the sound generation board and a toolkit of graphics objects for building user interfaces in a uniform style, while hiding implementation details and reducing development time.

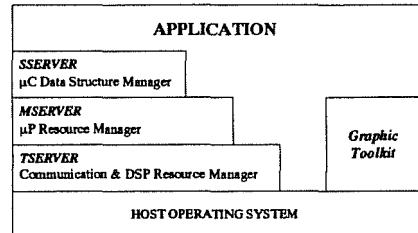


Figure 1: Application framework.

APPLI20 is the kernel of the EDIT20 architecture. In this sense, EDIT20 is the first complex and sophisticated example of an application program based on APPLI20.

APPLI20 architecture

APPLI20 consists of a set of three libraries that are in a hierarchical relation (SSERVER, MSERVER and TSERVER), and of a graphical toolkit that offers graphical objects in the EDIT20 style. A typical MARS application environment is shown in figure 1.

The APPLI20 architecture reflects the MARS data structure (see fig. 2), which can be divided into three “abstraction layers”:

- a. the Performance Environment,
- b. the Orchestra,
- c. the DSP.

The lower level (c) is related to the microprogram (μP), registers (REGS), data memories (DM) and sample memories

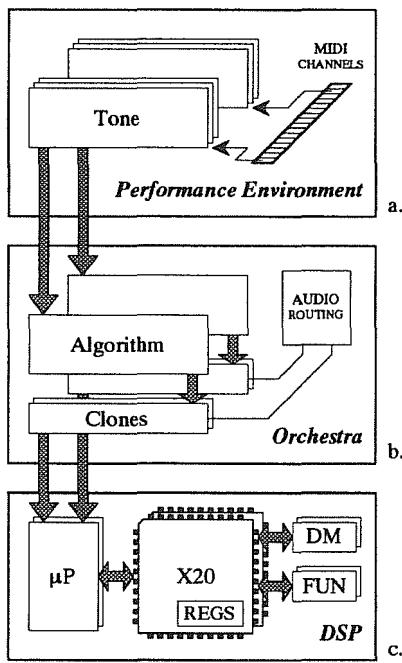


Figure 2: MARS data structure.

(FUN) of the two X20 DSPs. It also contains the virtual memory that maps the data memories of the two X20 chips to a single virtual space.

The Orchestra (b) offers a structured point of view of the DSP layer; it is a collection of algorithms described in terms of [1, 2]:

- number of voices (clones),
- entry points (envelopes, parameters, LFOs),
- audio routing.

The higher level (a) defines the MIDI Performance Environment, based on the Orchestra. It consists of tones

defined over Orchestra's algorithms, and of a Channel Map that links tones to MIDI channels. This environment is supported by the embedded real-time MIDI management (voice and tones allocation policy, events triggering, ...).

A tone is a complete configuration of an algorithm's variables. It assigns specific values and MIDI controls to the parameters of the algorithm. These are described in terms of:

- envelope definitions,
- values of static parameters,
- dynamic parameters' control structures and tables,
- LFO definitions.

APPLI20 libraries

T SERVER The Transmission Server implements access to the MARS DSP resources of the level (c), hiding from the user the details of data communication between the host and the board. The media for transmission can be chosen at run time between MIDI and parallel port. It is also possible to save on a file the "log" of a session for debugging or archiving purposes.

A summary of the functionalities offered by the T SERVER includes:

- initialization and test of the communication media;

- read and write of X20 registers (to perform ON, OFF, ...);
- installation of X20 micro-programs;
- direct read and write of algorithm parameters from/to X20 data memory (DM) locations, using physical or virtual addresses;
- load and dump of samples;
- pack and unpack of MARS messages.

DM locations are shared with the MARS realtime operating system which updates parameters in response to MIDI messages: it is the user's responsibility to avoid conflicts with his/her application.

MSERVER The Microprogram Server offers the programmer a set of objects and functionalities to build an Orchestra, hiding the details about level (c).

The Orchestra working environment consists of algorithms, each one composed of a collection of cooperating and interconnected objects: the modules. A module is the "atomic" object, and implements a portion of the microprogram code, allocating the necessary DM locations for its entry points (used for module interconnection and data entry).

The implemented set of functions includes:

- realtime creation and editing of algorithm modules (Add, Delete, Connect, ...);
- creation and editing of the Orchestra's environment as algorithm clones and their audio routing;
- transmission to the board of the X20's microprograms and data memory configurations using TSERVER;
- access to the Orchestra's data files in the EDIT20 format.

SSERVER The Structure Server extends the MSERVER objects and functionalities in order to allow the definition of a complete MIDI Performance Environment, level (a).

It offers an MSERVER-style approach to defining an Orchestra, adding special information for the MARS realtime control. All this hides the use of MSERVER, details of realtime requirements, and details of DSP aspects.

Moreover, it adds objects (dynamic and static parameters, LFOs, envelopes, data tables, tones) and functionalities (Create, Delete, Edit) to give values and MIDI control to an algorithm's entry points, to create libraries of tones and to link tones to MIDI channels.

SSERVER creates on the host an image of the three-level MARS data structure for

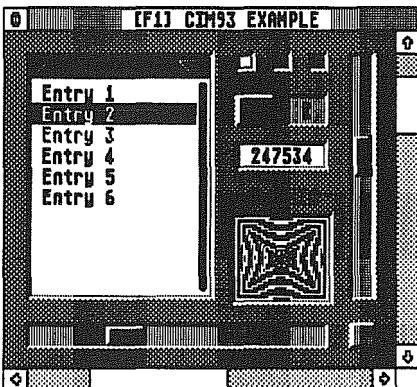


Figure 3: Graphic toolkit's objects.

a complete MIDI Performance Environment.

This image can be transmitted to the MARS board by means of high level object-specific functions that hide the use of TSERVER and the details of data protocol and I/O access. Furthermore, SSERVER offers special interaction modes to automatically transmit the image when modified.

The image is in EDIT20 compatible format. It can be saved on the host and retrieved for subsequent reuse. This means that an image can be exchanged between EDIT20 and other SSERVER applications.

Graphics Toolkit

The APPLI20 graphics toolkit is a set of libraries originally created to develop a MARS application in a graphical EDIT20 style [1, 2, 4]. However, it has no links with

the other libraries (T SERVER, M SERVER and S SERVER), and can be used to develop non-MARS graphics applications. Thus it was extensively used at IRIS to develop graphics simulators, editors and MIDI applications.

The toolkit allows the programmer to think in terms of graphics objects with callback mechanisms. The objects it manages are windows, list selectors, sliders, buttons, LEDs, editable text fields, and graphics areas (see fig. 3).

The toolkit's current implementation is based on the Atari GEM Operating System [6], using its AES (Application Environment Services) and VDI (Virtual Device Interface) functions to manage the user interaction with the objects in an event-driven fashion.

In order to allow MARS applications to easily migrate to non-Atari platforms [5], a great effort is currently being made at IRIS to evolve the toolkit to a portable version. This is being accomplished using multi-platform libraries (Cicero C-Tools, XVT toolkit) to reimplement the low-level layer of the APPLI20 graphics toolkit.

Examples

The following IRIS applications show how the APPLI20 libraries are employed to build a user interface and to con-

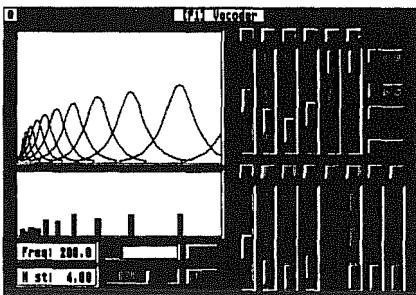


Figure 4: VOCODER.

trol and monitor in realtime the MARS board.

VOCODER This application represents a 12 channel vocoder, based on two banks of second order filters, where the frequency, the bandwidth, and the amplitude of each filter are controlled in realtime. An incoming signal from one ADC channel (typically a voice) is analyzed by the first bank of filters. The analysis results are used by the second bank for the resynthesis of external signals coming from another ADC channel. It is also possible to use a noise generator and/or a MIDI controlled oscillator as excitation sources for the resynthesis.

This application employs the graphics toolkit's callback mechanism to link user interactions with the activation of the other APPLI20 libraries.

First, the program initializes the MARS board, using MSERVER and SSERVER functions to install the MIDI performance environment pre-



Figure 5: CORDAM.

viously created by means of EDIT20.

After that, the graphics toolkit is used to manage user actions (slider movements, data entry, ...) while TSERVER functions are continuously called to monitor the amount of energy in each VOCODER's channel.

CORDAM This application implements the physical model of a string composed by 37 springs and 36 masses.

It is an experimental environment that lets the user configure the system, excite it, graphically visualize the results and hear them.

The parameters of the model can be changed in realtime while the string is playing. It is possible to apply an extra instantaneous force at any point of the string, by simply clicking the mouse at that point. One can also choose the strength and the direction of the force.

The algorithm computation, at the MARS sampling rate, and the slower redrawing of the string, are asynchronous. To synchronize them, a step-by-

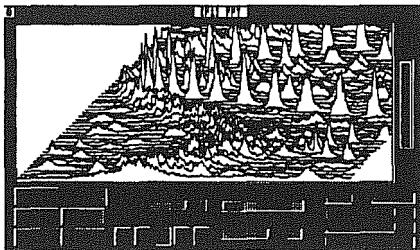


Figure 6: FFT.

step mode is also provided that makes easier to analyze the behaviour of the system.

The following parameters are configurable: 36 independent masses damping coefficient, elastic coefficient of the springs and gravitational acceleration. It is possible to free/lock independently the left and right edges of the string. The produced sound can be sampled and saved in a file to be used for pedagogical purpose.

The CORDAM's microprogram is written using the ASM20 assembler [3] to maximize the number of springs and masses in the string. EDIT20 is used to create an orchestra from Assembler programs. Only one X20 is used.

The orchestra is loaded using MSERVER; the updating of the parameters and the step-by-step mechanism are made through TSERVER calls. In order to edit the 36 masses, the user can specify his/her own formula that is translated and evaluated by an interpreter included in APPLI20.

FFT This application performs a 2048 band realtime Fast Fourier analysis on an input signal and graphically shows the results on the screen.

The input signal may come from the outside via the ADC converters, or may be synthesized by any algorithm running on MARS itself. In this case the FFT can be used to debug a previously defined algorithm.

The FFT is based on a special X20 microprogram, written using ASM20: it divides the input flow in 2048 sample slices, performs the analysis which gives a complex result, and computes the modulus and the logarithm of the modulus of this result; the output is saved at a fixed address in the X20 FUN memory.

The FFT program configures the DSP chips through TSERVER functions that load data and microprograms into the sound generation board. Then in realtime it reads data from the X20 FUN memory, displays them on the screen and manages the user interaction.

The graphics interface is built using the APPLI20 tools. The callback mechanism allows one to separate code for the user interface from that for graphics. FUN memory data is read continuously by the TSERVER functions.

Conclusion

MARS is an open system that can be used to explore and be adapted to a wide range of user needs.

APPLI20 has proven to be a powerful tool to easily and quickly develop user applications.

At this time, IRIS has developed applications in many fields, such as:

- *Music* – composition, live electronics, recording;
- *Education* – musical instruments' physical modelling, synthesis methods;
- *Science* – voice encoding/decoding with LPC methods, image processing.

IRIS is studying the portability of APPLI20 and, more generally, of the MARS system to other platforms [5].

Acknowledgements

Special thanks to Emmanuel Favreau who contributed to the APPLI20 design, and to Lorenza Bizzarri and Andrea Paladin who solved the DSP aspects of the CORDAM and FFT applications.

References

- [1] P. Andrenacci, E. Favreau, N. Larosa, A. Prestigiacomo, C. Rosati, S. Sapir:

"MARS: RT20M/EDIT20 – Development tools and graphical user interface for a sound generation board", ICMC '92 Proceedings, pp. 340-343, Oct. 1992.

- [2] F. Armani, L. Bizzarri, E. Favreau, A. Paladin: *"MARS-DSP environment and applications"*, ICMC '92 Proceedings, pp. 344-347, Oct. 1992.
- [3] E. Favreau, A. Prestigiacomo: *"ASM20 User's Guide"*, IRIS, 1990.
- [4] E. Favreau, S. Sapir: *"La Stazione MARS: dalla progettazione di algoritmi alla realizzazione di ambienti esecutivi dedicati"*, X CIM Proceedings, Milano 1993.
- [5] E. Maggi, A. Prestigiacomo: *"Portability of the MARS System"*, X CIM Proceedings, Milano 1993.
- [6] D. Prochnow: *"The GEM Operating System Handbook"*, Tab Books Inc., 1987.

LA STAZIONE MARS: DALLA PROGETTAZIONE DI ALGORITMI ALLA REALIZZAZIONE DI AMBIENTI ESECUTIVI DEDICATI

E. Favreau, S. Sapir

I R I S S.r.l.
Parco La Selva 151
03018 Paliano (FR) ITALY
Fax +39 775 533343 - Tel +39 775 533441
E-mail: mc2842@mclink.it

Abstract:

The Musical Audio Research Station (MARS) is a specialized digital machine for real time audio applications which has been entirely developed by the Italian Bontempi-Farfisa research institute IRIS (Istituto di Ricerca per l'Industria dello Spettacolo) located close to Rome.

MARS has been conceived as an interactive and integrated environment for audio research, musical production and computer music pedagogy. It is dedicated to people who have hit the limits of the digital musical instruments available right now on the market and to people who like a programmable and flexible sound machine with real time performance. The easy and interactive user interface provides a means of graphic definition for audio objects and an immediate sound feedback.

MARS is a development system for every type of real time digital signal processing such as analysis, synthesis, every type of filters, and sound effects. MARS is also a development system for sounds and MIDI environment that allows musicians to use it as a musical instrument, once configurated, such as any MIDI equipment of a musical studio.

1. INTRODUZIONE

MARS è costituito da una scheda di generazione suono chiamata SM1000 [4]. Questa scheda contiene:

- un microcontrollore (MC68302) per la gestione in tempo reale di tutto il sistema;
- due processori X20 per il trattamento del suono in tempo reale.

La potenza di un processore X20 può essere stimata valutando il numero di operazioni DSP eseguite in parallelo ad una data frequenza di campionamento. Un X20 può ad esempio realizzare un banco di 256 oscillatori indipendenti controllabili in frequenza ed ampiezza, oppure un banco di 256 filtri del secondo ordine. Ma può anche realizzare una FFT su 2048 punti, oppure 16 voci FM a 4 operatori o 16 harmonizer, oppure una combinazione di qualsiasi tipo di algoritmi.

La scheda SM1000 è collegata ad un personal computer tramite una linea MIDI e/o una linea parallela. Quest'ultimo supporta l'interfaccia utente per la configurazione di tutto il sistema.

Una volta configurata, la scheda SM1000 può essere sconnessa dal personal computer host, diventando così uno strumento controllato da MIDI.

Il software [1], [6] che consente la configurazione e l'uso della stazione comprende:

- il sistema operativo RT20M residente sulla scheda SM1000, incaricato della gestione dei processori X20, dei messaggi MIDI e di altri eventi asincroni;
- l'interfaccia utente EDIT20, ovvero l'ambiente di sviluppo grafico integrato con editors interattivi, usato per la progettazione di algoritmi

(processi di elaborazione del suono), **timbri** (configurazioni dei parametri di controllo degli algoritmi), **orchestre** (insieme di algoritmi) ed **ambienti esecutivi MIDI** (insieme di timbri e la loro mappa MIDI).

EDIT20 consente lo sviluppo immediato di tutti gli oggetti sonori e musicali che contribuiscono alla definizione di un ambiente esecutivo. Esiste, per ogni oggetto, uno strumento o un editor che ne permette la definizione.

Nei prossimi paragrafi verrà descritta una sessione di lavoro con EDIT20 che ne illustrerà gli aspetti più significativi.

2. ALGORITMI

Si vuole realizzare l'algoritmo di Karplus-Strong [7] che simuli le corde pizzicate. Per realizzare un algoritmo è necessario attivare l'editor ALGO. La videata corrispondente a questo editor comprende un menu, una zona di disegno e una palette di icone che rappresentano i moduli predefiniti dal sistema per la costruzione degli algoritmi.

EDIT20 fornisce circa 100 moduli che coprono la maggior parte dei bisogni nel campo DSP [2]. Questi moduli sono ordinati secondo criteri funzionali.

- Operatori elementari: aritmetici, logici, meccanici, e accessi alla

memoria esterna per campioni o linee di ritardo, ...;

- Generatori di segnale: oscillatori semplici, generatori di rumore, algoritmi più complessi come per esempio la sintesi additiva, ...;
- Generatori di segnale sotto campionato: inviluppi, LFO, vibrato, ...;
- Modulatori e distorsori: FM, DNL, AM, ...;
- Trasformatori logici: trigger, Sample&Hold, noise gate, ...;

• Effetti e trasformazioni applicati sul segnale audio nel tempo (ritardo, riverbero, ...), sullo spettro (filtri), sull'altezza (harmonizer, pitch follower, ...) e sull'ampiezza (mixer, pan, envelope follower, ...).

Per costruire l'algoritmo, l'utente deve assemblare nell'area grafica una serie di moduli interconnessi fra di loro. Questo viene realizzato, in modo interattivo, con l'ausilio del mouse e di combinazioni di tasti.

L'algoritmo è implementato sul processore X20 al momento della realizzazione grafica - senza tempo di attesa o di compilazione.

Nel nostro esempio (Fig. 1) si può osservare l'algoritmo KS. Un generatore d'inviluppo (env) controlla l'ampiezza del rumore (RND) per creare l'eccitazione della corda. Questa eccitazione viene poi inserita in una catena con

feedback costituita da una linea di ritardo e seguita da un filtro passabasso del primo ordine per simulare la propagazione smorzata dell'onda lungo la corda.

I moduli di questo algoritmo sono controllati da parametri i cui valori possono essere definiti molto facilmente durante lo sviluppo dell'algoritmo per verificarne il buon funzionamento.

3. DEBUG

EDIT20 mette a disposizione, nell'ambiente ALGO, un insieme di strumenti per il test dell'algoritmo che consentono di sentire e visualizzare il segnale. In effetti esistono 4 tipi di sonde che possono essere inserite in qualsiasi punto dell'algoritmo per seguire il flusso del segnale audio:

- 4 sonde audio rappresentate da un alto-parlante;
- 4 display numerici rappresentati da una piccola finestra quadrata;
- 4 tracce di un oscilloscopio rappresentate da una sonda;
- 4 ingressi per l'iniezione di segnale analogico rappresentati da una siringa.

Inoltre esistono altri strumenti comuni a tutti gli altri editors:

- VALUATOR è uno strumento per l'editing dei valori numerici. Ha la particolarità di poter rappresentare

e manipolare i dati nell'unità che si desidera (Hz, dB, ms, ...).

- X-DEBUG è uno strumento che simula una tastiera MIDI. Inoltre, consente di monitorare il contenuto delle memorie dati dei processori X20.

- WAVE-LOADER è uno strumento per accedere alla memoria esterna dei processori X20 per operazioni di dump o load di campioni su o da file.

4. TIMBRI

L'algoritmo KS visto precedentemente è controllato in modo statico con valori costanti. Per poter usarlo in un contesto musicale e dal vivo, è necessario trasformarlo in uno "strumento musicale".

Questo avviene definendo l'interfaccia timbrica dell'algoritmo. Ovvero scegliendo l'insieme dei parametri che devono essere "suonati" dal vivo, e stabilendo le regole di conversione da applicare sui dati ricevuti dai dispositivi MIDI di controllo.

In questo esempio (figura 3), si controlla la frequenza del suono cioè il parametro che fissa la durata del ritardo ovvero la lunghezza della corda.

Si crea quindi una variabile di nome *pitch* che sostituirà la costante del ritardo, fissata a 4 ms

nella figura 1.

La variabile *pitch* assume l'attributo DYNAMIC, cioè dipendente dai valori di dispositivi MIDI secondo una regola di conversione scelta fra i 17 preset.

In questo caso la regola di conversione è:

$$pitch = M_1$$

con M_1 definito come segue:

$$M_1 = KSPITCH(key) * 1.0$$

Con questa regola l'utente fa dipendere il valore del ritardo dal valore del tasto premuto. In effetti il valore del tasto (key) è usato come indice di una tabella (KSPITCH) che contiene la descrizione della scala di altezza desiderata.

Un timbro di questo algoritmo è quindi caratterizzato dalla definizione della variabile *pitch*, per quanto riguarda la sua risposta ai controlli MIDI, e dalla definizione dell'inviluppo *env*, per il suo andamento temporale. In questo modo per produrre diversi suoni basta definire dei timbri diversi sfruttando un solo algoritmo.

Gli editor PAR, ENV, LFO e TAB intervengono nella definizione del timbro per fissare rispettivamente i parametri di tipo DYNAMIC, gli inviluppi, gli LFO e le tabelle di conversione (come ad esempio KSPITCH)^[8].

5. ORCHESTRE

L'orchestra è costituita da un insieme di algoritmi, ognuno dei quali è collegato al sistema di bus audio (8 uscite e 4 ingressi). Per ogni algoritmo è necessario fissare il grado di polifonia, cioè il numero di duplicazioni dell'algoritmo. Così vengono definite le famiglie di algoritmi dell'orchestra.

In questo esempio (figura 4), l'algoritmo precedente KS è stato duplicato 8 volte, e la famiglia risultante è stata collegata al bus di uscita 1. Inoltre sono stati aggiunti 2 effetti, un effetto monofonico (EFFECT_1) e uno stereofonico (EFFECT_2). Questi ultimi algoritmi riprendono il segnale prodotto dalla famiglia KS sul bus 1, lo trasformano e lo diffondono rispettivamente sui bus di uscita 2, 7 e 8.

L'orchestra è realizzata tramite l'editor ORCH. In questo ambiente le interazioni con la scheda SM1000 non sono più eseguite in tempo reale, ma l'orchestra richiede una compilazione prima di essere salvata ed inviata alla scheda di generazione suono.

Esiste nell'ambiente ORCH un mixer che consente, dopo aver inviato l'orchestra, di bilanciare in tempo reale i livelli di uscita dei bus audio.

6. AMBIENTI ESECUTIVI

Per completare la sessione di lavoro con EDIT20, va introdotto il concetto di TMAP che è l'equivalente delle mappe MIDI presenti in tutti gli strumenti MIDI commerciali. La TMAP è descritta tramite l'editor omonimo. Essa raccoglie, per ogni algoritmo, l'elenco dei timbri utilizzati nella performance, e la relazione di default fra i canali MIDI ed i timbri.

La mappa MIDI può contenere fino a 128 timbri che possono essere assegnati dinamicamente ai canali MIDI durante la performance con l'invio di messaggi di *Program Change*.

In questo esempio sono stati richiamati 2 timbri per l'algoritmo KS: *Plucked* e *Bowed*. Il primo riproduce un suono di corda pizzicata mentre il secondo riproduce un suono di corda strofinata con l'archetto (la differenza dipende soprattutto dalla forma dell'inviluppo). Questi timbri sono stati associati rispettivamente ai canali MIDI #1 e #2.

Anche gli effetti hanno i loro propri timbri: *Effect_a* ed *Effect_b*. Questi timbri sono stati associati ai canali MIDI #3 e #4.

Una volta fissata la TMAP, l'utente dispone di un'orchestra multi-algoritmica pronta per essere usata dal vivo tramite MIDI usando una

master keyboard, un sequencer, o qualsiasi programma o dispositivo di generazione di dati MIDI. In questo esempio, l'utente potrà attivare gli effetti con un NOTE ON sui canali #3 e #4, e potrà suonare l'algoritmo KS in un contesto polifonico (8 voci) e multi-timbrico (2 timbri).

7. CONCLUSIONE

Con EDIT20 è possibile sviluppare in modo semplice ed intuitivo algoritmi, timbri, orchestre e mappe MIDI. Ma EDIT20 rimane uno strumento di uso generale che non ottimizza le risorse della stazione. Assieme alla stazione MARS vengono quindi forniti un assembler [5] ed una libreria "C" [3] per chi volesse costruire orchestre ottimizzate e sviluppare proprie interfaccie utente oppure proprie applicazioni di controllo.

MARS è stato utilizzato in diversi centri di ricerca per lo studio di nuove tecniche di elaborazione del suono e per la didattica dell'informatica musicale. Ma questa stazione si è rivelata uno strumento molto gradito dai musicisti soprattutto per quanto riguarda la realizzazione di ambienti esecutivi per il genere "Live Electronics".

REFERENCES

1. P. Andrenacci, E. Favreau, N. Larosa, A. Prestigiacomo, C. Rosati, S. Sapir, "*MARS: RT20M/EDIT20 - Development tools and graphical user interface for the sound generation board*", ICMC proc., pp. 340-343, San Jose 1992.
2. F. Armani, L. Bizzari, E. Favreau, A. Paladin, "*MARS - DSP environment and applications*", ICMC proc., pp. 344-347, San Jose 1992.
3. P. Andrenacci, F. Armani, A. Prestigiacomo, C. Rosati, "*APPLI20: a development tool for building MARS applications with an easy to use graphical interface*", atti del X CIM, Milano 1993.
4. S. Cavaliere, G. Di Giugno, E. Guarino, "*MARS: X20 architecture and SM1000 sound generation board description*", ICMC proc., pp. 348-351, San Jose 1992.
5. E. Favreau, A. Prestigiacomo, "*ASM20 User's Guide*", IRIS doc. interna, 1990.
6. IRIS, "*Musical Audio Research Station - Guida per l'utente*", doc. interna, 1992.
7. K. Karplus, A. Strong, "*Digital synthesis of plucked String and Drum Timbres*", Computer Music Journal, vol.7, n.2, pp.

43-55, Summer 1983.

8. G. Palmieri, S. Sapir, "MARS - Musical applications", ICMC proc., pp. 353-353, San Jose 1992.

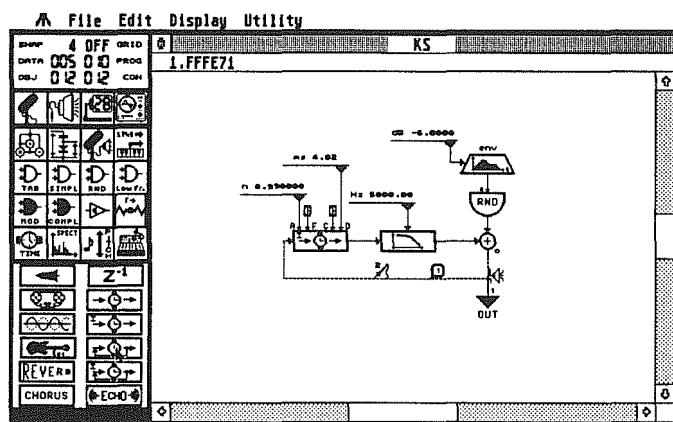


Figure 1. Videata dell'editor ALGO

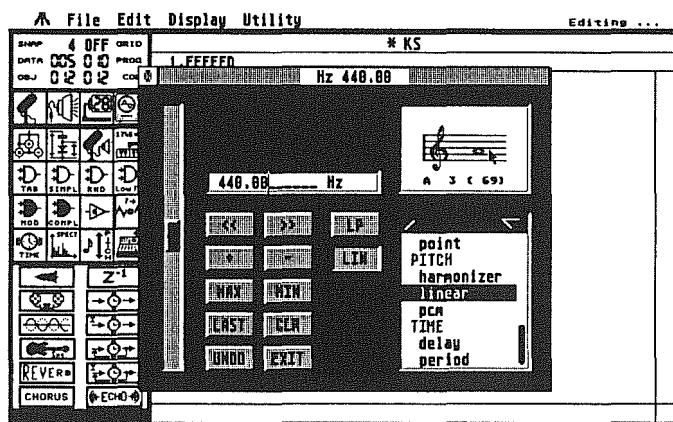


Figure 2. VALUATOR

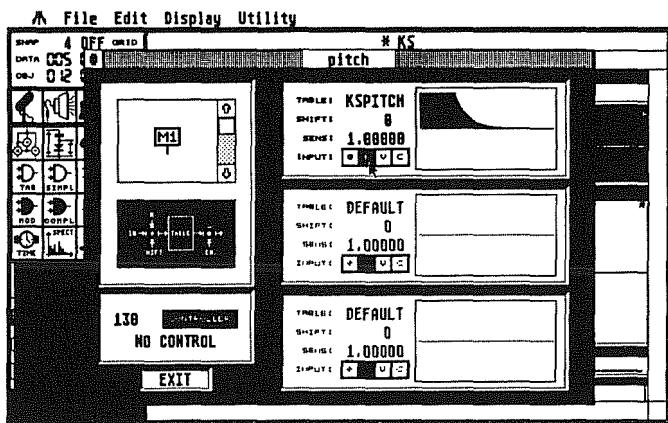


Figure 3. PAR: Definizione della variabile pitch

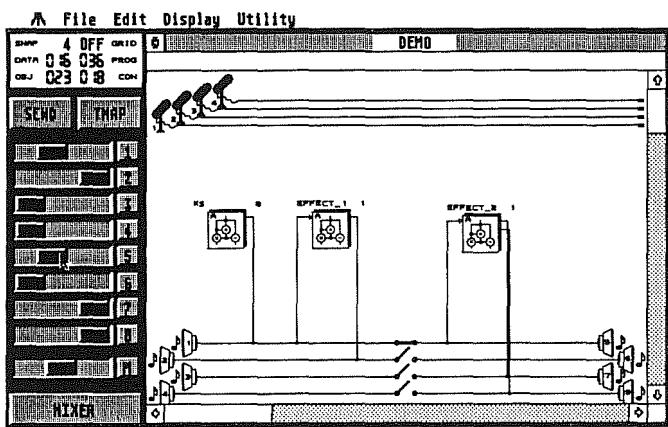


Figure 4. ORCH: orchestra DEMO

CELLE-FUNZIONE PER LA REALIZZAZIONE DI SISTEMI MUSICALI ELETTRONICI

E. Guarino, R. Bessegato, E. Maggi

I R I S S.r.l.
Parco La Selva 151
03018 Paliano (FR) ITALY
Fax +39 775 533343 - Tel +39 775 533441
E-mail: mc2842@mclink.it

Abstract

As for any kind of product the designers of musical systems must focus on the need for cost reduction, time to market, better performance, and consistent quality. These conflicting exigencies call upon the designer's ingenuity.

In the last decade silicon technology has moved through important and major improvements. It has been helped through computer-based design aids and libraries of pre-compiled functions.

But this is not enough: the trend toward more complex and complete systems has justified the rising use of dedicated functions that are eventually incorporated into the standard libraries.

At IRIS the need to implement families of musical systems has fostered the study and design of a dedicated library containing macrocells particularly suited for musical applications.

As a result IRIS has created a Microcontroller Softmacro, suitable for DSP management and control, and a DSP-like Sound Generator that can both manage a wide variety of sound-generation algorithms and implement a poliphony that is adaptable to the system's needs.

In this paper, utilizing IRIS's accumulated knowledge, the pros and cons of such an approach will be depicted, together with the description of a particular device designed according to these methodologies.

1. Aspetti evolutivi nella progettazione degli ASIC

La progettazione tradizionale dei circuiti integrati prevede il disegno a livello di transistor^{[1][2]}. Con questo metodo, oggi identificato con l'appellativo di "full custom", il progetto di un circuito integrato richiede notevoli risorse

umane e l'ausilio di sistemi molto costosi che solo le grandi compagnie di semiconduttori possono permettersi.

La grande diffusione degli ASIC (Application Specific Integrated Circuit), ovvero di circuiti progettati "su misura", ha potuto verificarsi soltanto dopo l'introduzione di un metodo alternativo, meglio noto come "semi-

"custom", che solleva la visuale del progettista dal transistor alla cella funzionale. L'approccio ASIC, spinto dall'esigenza di ridurre il costo progettuale, è stato reso possibile da alcune circostanze^[3]:

- l'evoluzione della tecnologia del silicio e la capacità di mantenere entro limiti sufficientemente prevedibili il comportamento dei transistor nei circuiti integrati di tipo digitale;

- la possibilità di rappresentare con semplici modelli astratti il comportamento di tali circuiti;

- la disponibilità di sistemi CAE abbastanza sofisticati da permettere di gestire la descrizione, il piazzamento e la simulazione di vasti insiemi di celle elementari.

La metodologia di progetto semi-custom possiede rispetto a quella full-custom la stessa valenza che ha un linguaggio ad alto livello rispetto ad un linguaggio assemblativo: minori tempi di progetto e superiore complessità a scapito dell'efficienza e della velocità. In particolare, per i circuiti integrati, risulta relativamente facile quantificare la convenienza dei diversi approcci: lo sforzo progettuale di un full-custom si giustifica solo se il numero di pezzi da produrre ne rende possibile l'ammortamento. Viceversa, produzioni limitate o ristrettezza dei tempi di sviluppo privilegiano l'agile metodologia ASIC, la quale a sua volta offre diverse alternative:

- l'approccio a Standard Cell (SC) ottimizza l'area e si presta all'integrazione di grosse celle caratterizzate da una struttura ripeti-

tiva (RAM, ROM, adders, multipliers, etc.) a fronte di un più elevato costo di sviluppo;

- gli approcci Gate-Array e Sea-of-Gates (GA,SOG) sono preferite in assenza di macrocelle (RAM, ROM) e/o in virtù del minor costo di sviluppo;

- la più recente soluzione Embedded Array (EA) rappresenta un compromesso tra SC e GA.

La tecnologia del silicio ha conosciuto negli ultimi anni uno sviluppo vertiginoso. Questo ha prodotto da un lato la drastica riduzione delle dimensioni dei transistor (di un fattore superiore a 10) con conseguente aumento della densità degli elementi circutiali e dall'altro la diminuzione del numero di difetti per unità di superficie che comporta l'aumento delle dimensioni possibili dei chip.

La combinazione di questi fattori ha permesso un notevole aumento di complessità dei circuiti integrati in generale e degli ASIC in particolare. Per meglio affrontare tale complessità, si capisce come sia diventato necessario un approccio logico-sistemistico che può ben definirsi "ad alto livello".

Queste circostanze impongono un uso massiccio di software per lo sviluppo e la verifica dei progetti: per citare qualche esempio, i simulatori analogici e logici, i modelli comportamentali, i pacchetti di sintesi logica e produzione automatica di Test-Vectors^{[4][5]}.

I sistemi musicali, sia nella parte di sintesi che in quella di controllo, non si sottraggono a

questa tendenza; allo stesso tempo l'obiettivo della massima integrazione rende consigliabile la presenza di sezioni "intelligenti" ovvero programmabili: queste, infatti, consentono la semplificazione del circuito attraverso l'eliminazione di complesse reti combinatorie e macchine a stati, oltre ad una maggiore controllabilità e linearità di progetto.

Un ostacolo a questa tendenza è tuttora costituito dall'alto costo di integrazione all'interno di un sistema di un qualsiasi processore commerciale. Può risultare allora conveniente persino il progetto di un processore costruito "ad hoc": questa soluzione risulta particolarmente vantaggiosa per sistemi di fascia bassa dove, malgrado la quantità dei controlli da effettuare, l'area occupata dall'unità di controllo deve essere ridotta ai minimi termini.

2. Distribuzione dei compiti in un sistema musicale

E' possibile individuare nei sistemi musicali elettronici alcune funzioni caratteristiche differentiate in base alla frequenza con la quale esse vanno processate. La realizzazione di tali funzioni si differenzia solo per le caratteristiche architettoniche, che rispecchiano i diversi vincoli a cui esse sono soggette. La sintesi del suono, ad esempio, deve avvenire in maniera rigorosamente sincrona, ripetitiva e veloce; i parametri a bassa frequenza (LFO, timers) sono anch'essi sincroni e ripetitivi

ma calcolati a velocità inferiori: infine, le interazioni del sistema con l'esterno (attraverso tastiere, switch o potenziometri) sono tipicamente asincrone e possono essere eseguite a velocità ancora inferiori, data la lentezza dell'intervento umano rispetto alla scala temporale di un circuito elettronico.

Da questa analisi, data l'assenza sul mercato di componenti funzionali dotati delle caratteristiche volute, deriva la scelta di realizzare un insieme di celle-funzione atte ad assolvere i compiti descritti, e che abbiano la caratteristica di essere flessibili, facilmente interfacciabili fra loro e scalari, cioè di poter aumentare o diminuire la complessità architettonale in funzione della complessità del sistema musicale nel quale vengono utilizzate.

Da un punto di vista realizzativo, una cella-funzione è un circuito costituito da celle elementari che realizza una determinata funzionalità di tipo logico a un livello di astrazione superiore.

La scelta di realizzare una libreria di questo tipo, se da un lato comporta lo svantaggio di dover impiegare un certo tempo di progetto e sviluppo, dall'altro dà il vantaggio di diminuire drasticamente i tempi di sviluppo dei prodotti che di tale libreria fanno uso, di conoscere a priori con una buona risoluzione il costo di ogni singola parte del sistema e soprattutto di modificare senza traumi la struttura del sistema in corrispondenza di variazioni delle specifiche.

che nel corso del progetto.

Vale la pena di sottolineare che l'integrazione di celle-funzione rappresenta una frontiera dell'attuale metodologia di progetto dell'hardware; è sufficiente a questo proposito ricordare che la CEE, nell'ambito del progetto ESPRIT, ha lanciato il progetto OMI-DE (Open Microprocessor Initiative Deeply Embedded) per l'integrazione di macrocellule con architettura complessa (RISC). In questo programma l'IRIS e' presente con il progetto MEDUSA^[6].

3. Descrizione di un dispositivo realizzato

Un progetto per il quale è stato adottato questo approccio è rappresentato da un dispositivo destinato alla realizzazione di un giocattolo musicale dai particolari requisiti di prestazioni audio (quali la presenza di una sezione ritmica, la politimbricità con varie tecniche di sintesi) e interattività con l'esterno (controllo di tastiera, playback, ecc.). Il prodotto doveva essere costituito da un singolo chip al fine di minimizzare il costo dei componenti e del loro assemblaggio. Dalle specifiche funzionali del prodotto e' stato possibile definire una struttura modulare per l'integrato, composta da un'unità di sintesi (DSP), un'unità di controllo (Soft Macro Controller - SMC) e una unità di conversione (PWM). Dato l'alto grado di integrazione adottato nel progetto, è stato scelto di cablare il software di controllo all'interno

del sistema. Questa soluzione, se da un lato offre i vantaggi di basso costo e minore necessità di intervento in fase di produzione, presenta il rischio di trasmettere a tutto il chip l'alta probabilità d'errore presente nel software. Per ovviare a questo problema, come si vedrà in seguito, sono necessarie procedure di test molto più approfondite delle consuete simulazioni circuitali.

Il DSP dispone di potenza di calcolo e lunghezza di parola adeguate al profilo medio-basso richiesto dall'applicazione. Ad esso è assegnata anche la parte di generazione dei segnali a bassa frequenza (LFO, timers) e quindi la sincronizzazione macrotemporale di tutto il sistema (metronomi, scansione della tastiera, controllo LED ed auto spegnimento).

Fra i compiti del SMC c'è la configurazione del sistema, la gestione del playback e dell'interazione con l'esterno, la trasmissione al DSP dei parametri necessari alla sintesi (frequenze, durate, timbri).

Alla cella PWM è affidato il compito della conversione DA.

Il dispositivo occupa una superficie di 16 mm².

3.1. La cella DSP

Il funzionamento del DSP si basa su un flusso circolare di dati sincrono che ruota attorno ad una memoria, controllato da un microprogramma. La memoria dati contiene le informazioni essenziali per realizzare la sintesi. La

semplicità della struttura non impedisce l'implementazione di svariati algoritmi per la generazione del suono: modulazione di fase, sintesi tabellare a due oscillatori interpolati, generazione di rumore, filtraggio, ecc.; altri semplici algoritmi realizzano un timer programmabile e un LFO. L'uscita audio digitale a 16 bit è inviata alla cella PWM.

Da un punto di vista funzionale il DSP consta dei seguenti blocchi fondamentali (fig 1): una memoria RAM di dati (DM) 32x16, un'unità aritmetico-logica (ALU) a 16 bit in grado di gestire automaticamente inviluppi di tipo ADSR, un moltiplicatore (MUL) 10x8, una memoria ROM 512x8 contenente le informazioni di timbro e le tabelle delle forme d'onda, una seconda memoria ROM (FREQ) per la conversione del codice tasto in frequenza; il flusso dei dati è regolato da una ROM di microprogramma (128x20), mentre una semplice interfaccia consente la comunicazione con la cella SMC.

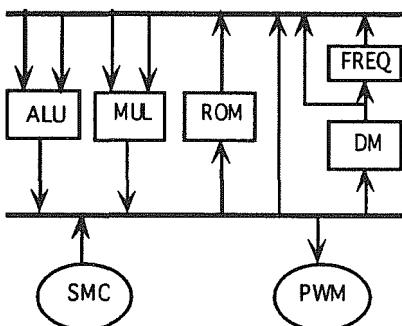


fig. 1

La gestione degli inviluppi, av-

viata da un'informazione di NOTON proveniente dal controllore, procede automaticamente al calcolo dei primi 3 segmenti dell'ADSR; il messaggio di NOTOFF attiva il quarto.

La dimensione del microprogramma è limitata grazie ad una opportuna organizzazione delle istruzioni in moduli: la combinazione dei vari moduli permette di implementare in maniera semplice, efficace e flessibile tutti gli algoritmi sopra menzionati. La modularità del microprogramma si rivela estremamente importante: essa consente di modificare le caratteristiche algoritmiche dei suoni senza influire sull'architettura del DSP.

Nell'implementazione in esame, la cella realizza una polifonia di 8 voci, ognuna delle quali può essere associata ad un algoritmo diverso.

La cella può essere facilmente scalata in funzione della polifonia e della precisione aritmetica necessaria.

3.2. La cella PWM

Questa cella realizza la conversione digitale-analogico a 11 bit senza penalizzare troppo dal punto di vista qualitativo l'uscita audio; allo stesso tempo essa riduce al minimo il costo del modulo di conversione. Infatti la sperimentazione acustica effettuata durante lo sviluppo ha confermato l'ideoneità della conversione PWM già a 10 bit, soglia facilmente raggiungibile con la frequenza di lavoro

del dispositivo (32MHz); trattando poi in maniera adeguata il bit meno significativo, si è elevata la soglia a 11 bit.

3.3. La cella SMC

La struttura di questa cella (fig. 2) risponde alle esigenze di completa controllabilità e di minimizzazione dell'area.

In particolare essa deve gestire la scansione della tastiera e dei pulsanti di comando, interpretarne il significato e trasmettere i parametri corrispondenti ai comandi ricevuti. Tra questi, la selezione del periodo di temporizzazione dell'intero sistema.

Una parte della gestione ha come effetto l'emissione immediata di messaggi verso la cella DSP, un'altra parte attiva la lettura e la decodifica di uno spartito, un'altra ancora ha il compito di ri-configurare il sistema.

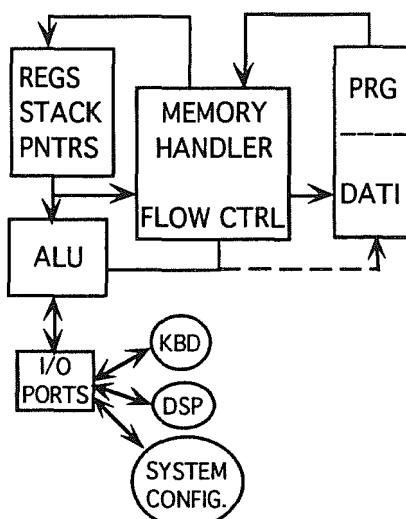


fig. 2

Nell'intento di risparmiare area, sono stati riuniti in un solo blocco fisico tutti gli elementi di memoria necessari:

- i registri di CPU;
- lo stack;
- i puntatori alla zona dati (utili per seguire individualmente le linee di polifonia dello spartito).

Il numero di tali elementi è stato determinato in funzione delle **particolari esigenze** dell'applicazione in:

- 32 registri (8 bit);
- 8 livelli di stack (16 bit);
- 8 puntatori (16 bit);

il tutto incluso in una RAM di 64x8.

La larghezza di parola scelta (8 bit) consente l'eventuale estensione dello spazio di memoria all'esterno del dispositivo.

Il set di istruzioni è stato scelto in base alle caratteristiche del sistema in cui la cella è inserita.

Ogni istruzione è realizzata mediante l'esecuzione di una serie variabile di nano-istruzioni: definite queste ultime, diverse sequenze generano diverse macro-istruzioni. La decodifica delle singole nano-istruzioni viene effettuata in due fasi, di cui la prima individua un gruppo omogeneo di nano-istruzioni, ovvero il sottogruppo di segnali di controllo coinvolti, e la seconda modifica effettivamente il loro stato.

Veloci modifiche del set di istruzioni possono dunque effettuarsi sia attraverso la riorganizzazione di una nano-routine, sia mediante un intervento sul se-

condo livello di decodifica.

Allo scopo di minimizzare il consumo e di sincronizzare il funzionamento del DSP e del SMC è stato inserito un meccanismo di "sleep", tramite il quale la cella può disabilitare il suo clock fino ad un evento di "awake" proveniente da un timer.

L'intera cella, esclusa la memoria di programma e dati, è composta da circa 2000 gates e può eseguire un'istruzione nel tempo medio di 160 nsec.

3.4. Il software per la cella SMC

Il software di base consiste in un linguaggio assembler^[7], realizzato con LEX e YACC su sistemi Unix, e in un simulatore scritto in C che realizza un modello funzionale della cella SMC.

Entrambi gli strumenti sono stati portati su macchine Atari data l'estrema facilità di interfacciamento di questi sistemi con piccole schede hardware.

L'ambiente ha soddisfatto l'esigenza di verificare la congruenza del set di istruzioni definito e la correttezza del programma applicativo da scrivere in ROM.

Successivamente, allo scopo di approssimare il comportamento reale del sistema, è stato realizzato un emulatore con circuiti FPGA (ACTEL) il quale ha permesso, unitamente al software di base, di collaudare il programma su un supporto hardware. Un altro emulatore permetteva la sintesi del suono e la messa a punto dei tim-

bri.

Una parte importante dell'ambiente di sviluppo è costituito da un pacchetto di software grafico altamente user-friendly per il monitoraggio e il debug dell'emulatore. Questo programma è stato scritto in modo da risultare portatile sia sull'ambiente software che sull'emulatore. Alcuni strumenti per la codifica degli spartiti e per la gestione della comunicazione MIDI sull'Atari hanno completato l'ambiente di sviluppo.

L'uso del simulatore C ha consentito di provare nuove istruzioni che, durante il test, sono state sostituite ad altre poco usate, con risparmio di memoria nel codice oggetto.

Il programma di controllo esegue la gestione di tastiera, l'allocazione delle voci nel DSP, il controllo dell'esecuzione degli spartiti ed alcune funzioni ausiliarie (gestione di LED, metronomo, controlli di volume e timbro). Inoltre la comunicazione con il DSP consente di regolare la velocità dei sincronismi e la frequenza del metronomo.

Un notevole vincolo per il programma di controllo è stato rappresentato dalla dimensione massima della ROM: 4k bytes, metà dei quali dedicati alla memorizzazione degli spartiti in dotazione allo strumento.

Il progetto della cella è progredito di pari passo con la definizione del software di base e dei primi ambienti di sviluppo software necessari alla stesura dei programmi. Questo sviluppo

parallelo incontra tre esigenze:

- poter già disporre del software quando il progetto hardware è terminato;
- poter simulare in maniera più accurata il progetto, attraverso la virtualizzazione fornita dall'ambiente di sviluppo;
- poter disporre di un feed-back continuo sia nella scrittura del software che nel progetto dell'hardware.

4. Conclusioni

L'approccio delle celle-funzioni e del software associato al loro sviluppo comporta lo svantaggio della dilatazione del tempo di progetto, ma offre gli aspetti positivi della flessibilità, della semplicità di intervento e della riusabilità dei prodotti.

L'esperienza ha dimostrato la validità della metodologia e fa presagire come sviluppi l'identificazione e la creazione di nuove celle, l'aumento della flessibilità e scalabilità delle celle già esistenti. Ad esempio, il potenziamento della cella SMC è stato oggetto di una tesi di laurea^[8].

References

[1] C. Mead, L. Conway:
"Introduction to VLSI systems", Addison Wesley Publishing Company, 1985.

[2] L. Glasser, D. Dobberpuhl:
"The design and analysis of VLSI circuits ", Addison Wesley Publishing Company, 1985.

[3] L. Abbondandolo, V. Liverini: "La progettazione di chip dedicati ", Elettronica oggi, Settembre 1986.

[4] R. Libeskind-Hadas, N. Hasan et alii: "Fault covering problems in reconfigurable VLSI systems ", Kluwer Academic Publishers, 1992.

[5] D. Ku, G. De Micheli: "High level synthesis of ASIC under timing and synchronization constraints ", Kluwer Academic Publishers, 1992.

[6] N. Larosa, C. Rosati: "MEDUSA: a powerful MIDI processor ", X CIM Proc., 1993.

[7] A. Prestigiacomo: "AX User guide ", documentazione interna IRIS, 1992.

[8] P. Paravano: "Un processore integrato per applicazioni musicali ", Tesi di laurea in ingegneria elettronica, Università dell'Aquila, 1993.

PORABILITY OF THE MARS SYSTEM

E. Maggi, A. Prestigiacomo

I R I S S.r.l.

Parco La Selva 151, 03018 Paliano (FR) ITALY

Fax +39 775 533343 - Tel +39 775 533441 - E-mail: mc2842@mclink.it

MARS^{[1][2]} is a development system for real time audio applications. Its complex hardware and software architecture derives from the requirements of real time processing, interactivity, graphic interfacing, archiving and communication.

More than 100,000 lines of C code were written for the ATARI host to implement VME and MIDI communication with the sound board, the GEM-based EDIT20^{[1][2]} and to support the development of MARS applications.^[3]

In order to adapt MARS to the most common hardware and software platforms (Mac, UNIX, DOS, Windows) a project to study and improve the portability of MARS was initiated at IRIS. It mainly focused on the MARS host environment. The sound board encapsulates real time aspects and depends upon the host only for its parallel connection.

This paper will describe the results of that project with some reusability and maintenance aspects.

Portability Strategies

An application is portable across a class of environments to the degree that the effort required to

transport and adapt it to a new environment in the class is less than the effort of redevelopment.^[4] In MARS the portability will concern both the development environment and the applications.

Differences among the target platforms make binary portability impossible. In what follows, portability will be applied to sources, data and user skills.

The key problem of the designer is to choose portable interfaces to the resources used in the project, such as languages, libraries, operating systems, file systems, I/O devices, GUIs and networks.

Different yet compatible strategies may consist of: a) the selection of resources portable or standard or in some way available on target platforms; b) the adaptation (manual or automatic) of resources to target environments.

The first requires a deep analysis of products and strategies on the market. The second may involve automatic translation, dynamic run-time adaptation, and designing for portability.^[4] In this case, the designer has to identify and isolate the critical aspects that require adaptation, such as the MARS TSERVER^[3] and file manager. This can be done through the

design of new portable interfaces that abstract and encapsulate (or hide) non-portable aspects, and through a good modular structure that isolates non-portable code.

Even if no object-oriented methodology or tool is adopted, the design of interfaces should refer to some basic principles upon which object-oriented and, usually, good design is founded (abstraction, encapsulation, modularity, hierarchy)^[5]. It should consider metrics like coupling, cohesion, sufficiency, completeness, and primitiveness,^[5] in order to also deal with resource reusability and maintenance.

The following will describe typical critical aspects, also encountered in the development of MARS, together with IRIS's strategies and solutions.

Critical Aspects

- *The Programming Language* - It is the basic resource for portability and should hide details of any usable resource.

MARS uses the standard ANSI C, but no commercial compiler is portable on all target platforms and compilers on different platforms have different behaviours. They offer different development environments, but even worse they make different assumptions on issues not solved by the standard or they offer non-portable directives.

For example, compilers define differently the size of scalar types, like *int*, with possible problems on bit-mask or size-dependent

operations and on transferring data via a serial port or a binary file. Moreover, they have different rules for the size, alignment, and padding of structures with possible problems of program memory consumption and of incorrect access to structures and structure members. Directives, like *pragmas* and command-line flags, can be used to control these problems, but they may not be portable and are dangerous when used for *include* interfaces to run-time libraries compiled with different directives. New abstract interfaces should hide these aspects. Code should contain names rather than references depending upon the memory order and the data size.

In any case, ANSI C, like most programming languages, omits or inadequately specifies interfaces (Application Programmer Interfaces) to the other resources. The designer is thus forced to look for many other portable interfaces.

- *The Operating System (OS)* - It concerns functionalities like multiprogramming and process synchronization. In MARS, real-time aspects are dealt with in the sound board. The host environment uses few functionalities, like I/O, event handling or the *exec* of an external program, waiting for its termination. No standard or portable API is available on all target platforms, even if UNIX-based OSs and Windows-NT are going in this direction. So, in MARS this aspect was abstracted and isolated.

- *The File System* - It concerns operations like *CreateDir*, *RemoveDir*, *MoveFile*, *DeleteFile*. Some portable interfaces are available from portable GUI packages mentioned below, but in MARS a new abstract platform-independent interface was defined that includes the required file system functionalities.

- *The I/O System* - It concerns the access to I/O devices and in MARS, particularly, the transfer of data via MIDI or parallel lines. A transmission server, TSERVER,^[3] was designed to be easily extensible and portable to different platforms and transmission lines, through the abstraction of low-level access to devices and of data protocols. TSERVER supports application portability, because it offers a unique API that hides I/O device access, data protocol, and even the currently selected transmission line. In APPLI20^[3] TSERVER supports MSERVER and SSERVER portability. The I/O problem was also solved in the board.

- *Graphical User Interface* - GUIs like MS-Windows, Macintosh, OSF/Motif, are very popular, but no two GUIs are the same. Each has its own look and feel. Window styles, mouse actions, and menu arrangement all differ. GUI API's are still different, non standard, and non-portable on all target platforms.

The early adoption of the GEM/Atari platform for the MARS user interface is forcing IRIS to

rewrite all graphical code to reach a high degree of portability. The designer could create a platform-independent API that abstracts the functionalities of target GUIs, isolating and hiding from the application non-portable GUI aspects. IRIS made some efforts in the graphic toolkit of APPLI20^[3], but it is difficult to define an API able to satisfy all MARS graphical needs on all target platforms. This problem contemplates the creation of portable high level toolkits, event handling, graphics, graphical resources, and interface building tools.

However, support for portability is available on the software market. A lot of portable user-interface libraries and portable APIs are flooding the market, that support the portability of graphical applications on different GUIs across multiple platforms, such as XVT, Wndx, CommonView3, Cicero. The choice depends upon the level of support and functionality one requires, but there is certainly a loss of efficiency. The real problem is that none of them is standardized, even if XVT has been chosen by the IEEE committee as the base for drafting a Layered API for GUIs. For tools and graphic resources, each vendor adopts its own products. IRIS has the further problem of providing the portability for the ATARI/GEM platform, which is supported only by Cicero.

- *Data* - In MARS, this concerns data files describing algorithms,

timbres and orchestras that are exchanged among applications running on multiple platforms. The ASCII format helps portability, but disk space consumption and efficiency sometimes require a binary format. Rather than using conversion tools, it should be possible to read/write data across multiple platforms. This can be obtained abstracting memory dependencies (for example, swapped bytes for an integer on Motorola and Intel) and compiler differences (for example, integer and structure sizes and padding). In any case, code portions should be isolated and hidden from higher level users, when they concern file accesses and data formats.

- *Internationalization* - It concerns the portability of software across different languages. A programmer should not hard-code messages or language-dependent output text.
- *Experience* - This concerns the portability of the user skills across the target platforms. It is compromised by the differences between operating systems, development environments, file systems, and GUIs. A developer has to know more than one platforms, in order to recreate his/her tools (makefile, shell scripts) or even to reorganize the directories.
- *Other aspects* - Luckily for us, the problems of *networking* and *parallel processing* did not apply to the MARS project.

Conclusion

The lack of universal standards

makes it difficult to improve the portability of a complex system. In MARS, this is true especially as far as the graphical user interface is concerned. A part of APPLI20 is already running on more than one platform. Reusability and maintenance take advantage of and depend upon the design portability choices. Portability goes toward the definition of a virtual (abstract) computer. More help should come from Software Engineering to automate portability.

References

- [1] P. Andrenacci, E. Favreau, N. Larosa, A. Prestigiacomo, C. Rosati, S. Sapir, "MARS: RT20M/EDIT20 - Development tools and graphical user interface for the sound generation board", pp. 340-343, ICMC Proc., 1992.
- [2] E. Favreau, S. Sapir, "La Stazione MARS: dalla progettazione di algoritmi alla realizzazione di ambienti esecutivi dedicati", X CIM Proc., 1993.
- [3] P. Andrenacci, F. Armani, A. Prestigiacomo, C. Rosati, "APPLI20: A development tool for building MARS applications with an easy to use graphical interface", X CIM Proc., 1993.
- [4] J. D. Mooney, "Strategies for Supporting Application Portability", IEEE Computer, Vol.23, N.11, pp.59-70, 1990
- [5] G. Booch, "Object Oriented Design with Applications", The Benjamin Cummings Publishing Company, 1991

Sezione 6c

Altre workstation

MUST C25
STAZIONE DI LAVORO MUSICALE CON
SCHEDA DSP LEONARD'C25

G.Bertini, D.Fabbri, M.Marani, L.Tarabella.

Istituto Elaborazione Informazioni-CNR

Via S.Maria 46, I-56126 Pisa

fax +39 050/554342; e.mail:bertini@iei.pi.cnr.it

Abstract

MuSt C25 is a workstation for real-time audio musical signals processing and synthesis based on a low cost, modular multiDSP system (named MULTIC25) This system is based on a MSDOS compatible Personal Computer and a number (up to eighth) of DSP boards LeonardC25 (Leonardo Spa, Massa, Italia), carried out around the TMS320C25 microprocessor; the boards are connected in a daisy-chain configuration via the high speed serial ports provided by the TMS; the first and the last board are then connected to, respectively, an analog-to-digital and a digital-to-analog external circuitry.

The operational environment for the MuSt C25 has been developed as an application for Windows 3.1. The main functions of MuSt C25 are tasks distribution among the various modules and management of the flow of controls parameters and synchronisation. Tools for graphically editing and

compiling synthesis and analysis algorithms, for hard-disk recording operations and for integrating the MuSt C25 with DSP-MIDI environments are under development.

Introduzione.

Un sistema digitale programmabile atto a trattare segnali audio musicali, per avere un grado di polifonia e ricchezza timbrica tale da soddisfare le esigenze della didattica, della ricerca e dell'esecuzione in tempo reale, deve possedere potenze di calcolo dell'ordine di almeno decine o centinaia di MIPS. Per raggiungere tali livelli di performance con un buon rapporto costo/prestazioni, abbiamo proposto (l'IEI ed il CNUCE, Istituti pisani del CNR, sono da tempo attivi nel settore con realizzazioni di sistemi in tempo reale per la sintesi in tempo reale [1]) una soluzione che consiste nella realizzazione di sistemi paralleli opportunamente

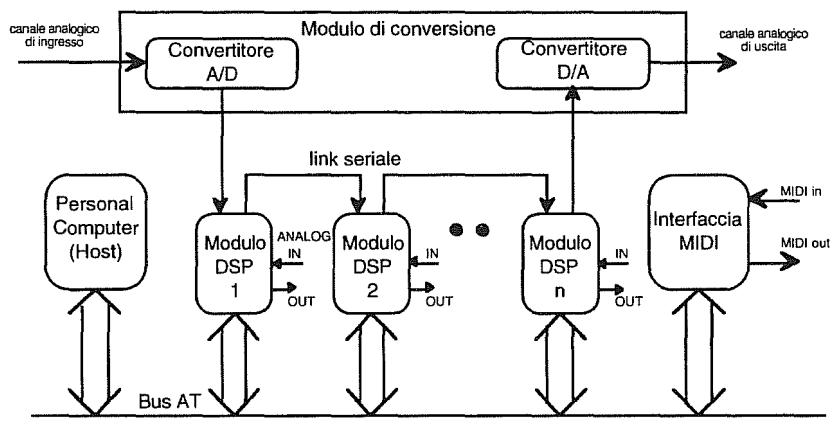
configurabili, basati su processori DSP programmabili di tipo commerciale e su elaboratori della classe personal [2].

Architettura della MuST

Per poter sperimentare le soluzioni proposte in [3] e sviluppare applicazioni con ridotti costi dell'hardware, è stato assemblato un sistema con architettura semplificata basata su bus AT [4]. La stazione di lavoro musicale 'MuSt C25' è composta quindi dal sistema MULTI-C25 [5], da un software di base e da alcuni tools applicativi. MULTI-C25 è un sistema modulare basato su schede LeonardC25 [6] (fino ad un massimo di otto) compatibili con il bus AT.

Le schede LeonardC25 utilizzano il microprocessore della Texas Instrument TMS320C25, banchi di memoria EPROM e RAM veloci; sono inoltre corredate di un'interfaccia analogica, circuiteria di condizionamento dei segnali e di una porta seriale ad alta velocità (5 Mbit/s), che implementa il protocollo di comunicazione DSP della Texas stessa.

Nel nostro sistema le schede sono collegate in configurazione daisy-chain, senza necessità di logica aggiuntiva, tramite la porta seriale suddetta ed il primo e l'ultimo modulo sono connessi ad un opportuno dispositivo di conversione A/D e D/A. Per quanto riguarda l'interfaccia MIDI è stato utilizzato il modulo commerciale (MPU-401 PC, della Roland).



Architettura della stazione

Sono in via di completamento dei tools software per rendere disponibili sul sistema funzioni di utilità come un editore/compilatore grafico di algoritmi di sintesi, funzioni di hard-disk recording e di integrazione di ambienti DSP e MIDI, ecc.). Il sistema opera essenzialmente nel modo seguente: con cadenza temporale pari al periodo di campionamento prescelto, tutti i moduli eseguono in parallelo gli algoritmi ad essi assegnati e trasferiscono i dati da un modulo a quello successivo per mezzo della porta seriale del TMS320C25. Le funzioni principali del software sono quelle di controllare il trasferimento dei dati e di gestire il passaggio dei parametri inviati da host per il controllo in tempo reale degli algoritmi in esecuzione sulle singole schede.

L'ambiente di lavoro.

L'ambiente di lavoro è stato sviluppato sull'Host, un calcolatore Ms-Dos compatibile, come applicazione per il sistema operativo Microsoft Windows 3.1: allo start-up il programma riconosce la configurazione hardware installata e carica il software di base sulle memorie dei moduli DSP; le funzioni affidate ai moduli possono essere decise dall'utente che può specificare il compito di ciascun modulo di calcolo scegliendo fra operazioni di sintesi ed elaborazione.

La finestra principale del programma consente l'editing contemporaneo di più file di testo che costituiscono il codice specifico relativo agli algoritmi da caricare sulle schede. Al momento questi algoritmi devono essere scritti in assembler TMS320 arricchito di alcune nuove istruzioni e direttive definite come macro, raccolte in un file di sistema. L'utente ha poi a disposizione un'ulteriore serie di macro definite in una libreria, contenenti le specifiche dei più comuni algoritmi di sintesi.

Per gestire i dati digitali provenienti dal convertitore A/D, sono disponibili alcuni strumenti che consentono di memorizzarli temporaneamente sulla RAM dell'Host. Dalla Ram possono essere quindi caricati sui moduli o trasferiti come file verso Hard Disk.

Come esempi applicativi che consentono il controllo degli algoritmi sono stati sviluppati due strumenti: un 'Sequencer' e una finestra di 'Controlli in tempo reale'.

Il Sequencer è strutturato ad imitazione dei registratori audio multi-traccia (così come i più comuni sequencer commerciali). La finestra di 'Controlli in tempo reale' consente all'utente di configurare una serie di cursori, manovrabili sia tramite il mouse, oppure tramite input esterni (segnali analogici acquisiti direttamente dalle schede, uscite di dispositivi MIDI, ecc.).

I particolari di progetto e una guida utente sono contenuti in [7]

Conclusioni

La stazione di lavoro si presenta come un sistema aperto, completamente programmabile, utilizzabile in diverse applicazioni, dalla progettazione di algoritmi di elaborazione e di sintesi di segnali audio e musicali, alla didattica e alla produzione musicale creativa. Essa è corredata di alcuni strumenti software per funzionalità tipiche del settore come il trattamento grafico di algoritmi di sintesi, l'hard-disk recording, interfacciamento ed integrazione verso il protocollo MIDI.

Dato il basso costo delle schede Leonard'C25 (0.5 MLit), la stazione rappresenta di fatto un sistema facilmente acquisibile per un primo approccio all'uso di supporti di calcolo per il DSP basati su personal computer ed è rivolta ad un'utenza tipicamente costituita da laboratori di scuole musicali e di ricerca, compositori e musicisti ecc..

Le funzionalità della stazione possono essere ovviamente valutabili solo in una seduta dimostrativa; le più significative finestre di interfaccia utente sono state inoltre riportate in un poster.

Questo lavoro è stato svolto con il contributo del Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo", Sottoprogetto 2, Processori Dedicati.

Bibliografia

- [1] G. Bertini, M. Chimenti, F. Denoth - "*TAU2: un terminale audio per esperimenti di Computer Music*", Alta frequenza Vol. 46, pp. 600-609, 1977.
- [2] E A.Lee - Programmable DSP Architectures -IEEE ASSP Magazine, Vol.5, n.4 pp. 4-19, Vol.6, n.1 pag. 4-14, 1988.
- [3] L. Tarabella , G. Bertini - "*A Digital Signal Processing System and Graphic Editor For Synthesis Algorithms*" ICMC Proceedings (Columbus, Ohio, USA) pp.312-315, 1989.
- [4] G. Bertini, L.M. Del Duca, M. Marani - "*The LeonardC25 System for Real-Time Digital Signal Processing*" Proc. Int'l Workshop in Man Machine Interaction in Live Performance, Pisa , pp. 107-117, 1991.
- [5] G. Bertini, D. Fabbri: "*MULTIC25, un sistema Multi-DSP con schede LeonardC25*" Rapp. Tecn. Prog. Finalizzato "SICP"-CNR, R/2/108, 1993.
- [6] G. Bertini, L. Tarabella, G. Bacchicocchi, M. Balestieri: "*Tecniche di gestione delle risorse in sistemi multiDSP per la sintesi e l'elaborazione di segnali audio*" Rapp. Tecn. Prog. Finalizzato "SICP"-CNR, R/2/55, 91.
- [7] G. Bertini, D. Fabbri, L. Tarabella: "*MuStC25, una stazione di lavoro musicale con il sistema MULTIC25. Manuale operativo.*" Rapp. Tecn. Prog. Finalizzato "SICP"-CNR, R/2/109, 1993.

CONSTRAINTS SATISFACTION PROGRAMMING IN COMPUTER AIDED COMPOSITION ON A HIGHLY GESTUAL DEVOTED SYSTEM, BASED ON A VME-MULTI-PROCESSOR JOINING TRUE UNIX AND REAL TIME

Philippe Prévot & Arnaud Debayoux

LIMCA (Lutherie Informatique et Musique Contemporaine à Auch)

Château de Saint-Cricq - Route de Toulouse

F-32000 AUCH - FRANCE

Tel +33 62.63.29.54

Fax +33 62.63.44.23

The objective of this work is not to start the dispute again between real time and deferred time, but on the opposite to tempt a conciliation approach. Besides, at the time when this article is written, the work is fully in progress.

From the coming out of specialized processors and DSPs, signal processing has become, often with quality loss, the prerogative of real time. As a matter of fact, the pleasure of being able to modify a sound in reaction to listening and without intermediate, still facilitating the work on sound material, often prevails on the will of very finely defining some subtle parts of the sound, like the attack or the polyphonic superposition conditions. The transition processes, so important in the perceived quality, are often neglected, because difficult to manage in real time. Reproducibility and stability,

which made up the glory of digital versus analog techniques, are never truly guaranteed but in deferred time, even though it doesn't exist any essential impossibility, but only big practical difficulties.

In fact, from the user's point of view, real time is an illusion, if one forgets that the freedom degrees at execution time, are limited to a class of sounds and musical schemes, previously defined in deferred time.

From the designer point of view, real time is far from just going as fast as possible, but should:

- guarantee the delays as deterministic as possible.

- be able to measure these delays, and to control them by controlling the amount of processes.

- allow asynchronous task execution, by an essentially non-linear architecture.

Besides, real time work practice is nothing but a succession of real time work sessions, constituting by themselves a deferred time work session, compared to some "final execution".

In terms of time structure building, the computer aid to composition is still the privileged area of deferred time, for three possible reasons:

- the necessary reflection time for composing often makes real time unnecessary.

- the complexity and the huge diversity of the problems, and of the ways of solving them, imply that the solutions may hardly be given by a unique software, and are often processed in languages like Lisp or Prolog which are not well suited for real time execution. Their real time processing may be difficult, independently from the language used.

- some problems, among the most interesting ones, may imply a non monotone time processing: it is possible that:

$$x(t_1) = f(t_1, y(t_2)) \quad \text{where } t_2 > t_1 \text{ and where } y(t_2) \text{ may not be known at } t_1.$$

Such an indictment against real time should terminate the discussion.

And yet the last years have been marked by an increase of the DSPs computational capacity, by an enhancement of the sound possibilities which result in increasing the number

of simultaneous voices and parameters that one system has to hold at a given moment, and also by the increasing power of general computers. A computer music instrument maker should feel guilty not to try to merge real and deferred time. The finest job to give life to a sound can hardly replace the pleasure of playing an instrument in live music.

Real time never exempted the composer to compose, would it be by a "top-down" pyramidal genesis, by building a structure from the sound material, by non-arborescent nets, or by any other method which make the conception time be neither monotone nor linear versus musical time. Finally real time, beside the technical questions, could be defined by three rules:

- 1- it is submitted, more or less, to an unpredictable external world (environment rule).

- 2- it renounces to the non-monotony of its execution time versus the musical thought time (monotony rule).

- 3- it should guarantee the linearity, in a 1 to 1 rate, of the execution time to the desired musical time (linearity rule).

We daily develop the first rule. We do not have (yet) tried to transgress the second one. We do try to get round the third one.

System architecture

This is why we designed a VME based multi-processor system, comprising:

- a DSP board "Quatron" featuring up to 64 wave-shaping generators, or 128 table-lookup generators, with 128 ramp generators, and 128 timers, at a sampling rate ranging from 27 KHerz to 3.5 MHerz
- driven by a 68030 real time board,
- and a 68040 UNIX board, under X11,
- various interface boards for gestual control devices, among which midi,
- and a gestual console - Pacom - featuring 32 potentiometers, 8 incrementals, 76 push buttons, 176 leds, 14 8-character displays, under midi communication.

None of the elements above is of a brand new design, but are still above all commercial devices.

There are two reasons for a double-processor:

- being opened to the Unix world, and able to use high level libraries from any source.
- sharing the work between a relatively large time scale under Unix on one hand, and real time on the other hand. Unix manages the displays under X, as well as some events like mouse clicks, for low communication flow interaction. The Real Time Processor (RTP) manages the sound process details, as long as

time allows it, as well as the midi communications. The "Constraints Satisfaction System" (CSS) is run by Unix.

Both processors share a common memory, allowing a dynamic software link, without delay, but can also be used independently on their own disks and displays.

The fact that the DSP is mapped on the bus allows a non-delayed control of all the audio parameters.

The midi interface board allows to extend sound processing to any midi external instrument.

How to handle it ?

Such a system, according to the user's specific patch, may typically present more than a hundred different parameters to handle (for example, up to 15 parameters for 10 independant instruments, inside the Quatron). Even some parameters, classically considered as a one parameter like wave tables, should be replaced by a set of parameters, if not defining exhaustively the objects, at least defining one or several functions allowing to classify the objects in a set.

Facing such a large number of data to hold, the musician wants to concentrate himself, in a given situation, on certain variables, and expect from the computer to temporarily hold the others, in the "best way", that is by satisfying, as well as

possible, a set of equations and unequations, logic or arithmetic, linking them to each other or to some parameters, imposed from elsewhere.

These techniques, known under the general "Operations Research" methods, have been of high interest for computer researchers for a long time. But it generally lead to solve problems by or for languages such as Lisp or Prolog, which are not clearly designed for real time.

On the opposite, real time implies that the constraints cannot, in a binary way, be or not be satisfied - since music has to proceed anyway (monotony rule)- but that the solutions must push towards an ideal value. Thus we must introduce a compromise:

1- the result of each constraint is not a definitive value but a temporary target to reach.

2- a wrong or too slow direction is not fatal, but itself generates its own correction.

Definitions

At this point, let us define a few elements:

The *external world* consists of all sorts of devices giving unpredictable values to predefined objects, thus creating unpredictable events attached to predefined functions. They typically are potentiometers and buttons from the Pacoms, midi events, widgets events from X11, but also interrupts and values read from the DSPs, or

predefined values from a score, instantiated by some external event, etc... The external elements give the *control* values.

The *parameters* are elements the value of which can change in time, independently from the CSS, and are what the CSS sees of the *external world*. They receive their values from the *controls*. Their mobility keeps the constraints from being definitively satisfied. They are read only by the CSS and written by the RTP.

The *variables* designate the elements of which the value is fixed by the CSS after satisfaction. They are read only by the RTP and written by the CSS.

Where to start ?

Starting from the patch definition, the user defines the *parameters* to be communicated by the RTP to the CSS, both in terms of an algorithm based upon different *controls* and by a computation instantiator, that is either a regular rate or an external unpredictable reason to deliver a new value.

The user then defines the *variables* to be expected from the CSS and the different points in the patch where to use their values. The system state at a given time t is the set of the *variables* values:

$X(t) = (x_1(t), x_2(t), \dots, x_n(t))$.
The set of *constraints* are then defined, in different ways,

according to their type.

Where to go ?

For given values of the *parameters* at a given time t , a *constraint* C_i defines a subset $\zeta_i(t)$ of "ideal" points. A *pole* is the association of a *constraint* and a sign: repulsive or attractive. It exercises a *strength* of that sign, in the direction of the shortest way between $X(t)$ and $\zeta_i(t)$, the magnitude depending on the way length. The *poles* are weighted to differentiate a priority, relative to each *constraint*. The weights vary with time. This is our first way of getting round the linearity rule: according to the necessity at a given time, we do not affect the same computational power to each *variable*.

From the weighted strength sum results a global *strength* which is going to displace the system to a new state, or *target*, $X(t+\delta t)$ where δt is the CSS sampling period. These successive *targets* are taken into account by the RTP. As δt can be relatively large if the *constraints* are complex, the RTP processes a series of linear interpolations between the effective state and the present *target*.

Strength computation

The *strength* may be different according to the distance sign, that is the *pole* side. For an unequation, it is

null on one side. The most interesting one seems to be increasing with the distance, insuring the system self-limitation. Furthermore, the strength weighting before application of the global *strength* allows to take into account the shifts accumulated by a *constraint*. In this way, if a *pole* comes to trap the system, it rapidly looses its relative strength and frees the system back. According to the way the *constraint* is defined, the *strength* magnitude and direction are computed by the gradient or by the orthogonal projection.

A *constraint* may be defined:

- as a linear or non-linear function of n variables.
- as a set of definitions on discrete definition intervals.
- as a Hermite cubic, by a set of points and derivatives.
- or in any way, through a C-code giving the signed distance to the hyperplan and the partial derivatives.
- (in progress) as a logical function, like belonging to a set, etc...

Different *constraints* may be gathered in a *character*. The user can create a *character* library and build his musical mechanisms as a succession or superposition of *characters*.

Problems

Real time is still the first and last constraint to be satisfied. If not, the constraints satisfaction

fineness should be cut. A hardware timer measures the evaluation process charge by its response delay. The step of any iterative method and the transition step to the target state increases in ratio to this delay. The transition and satisfaction processes are asynchronous, and they share the processor time depending on the delay. Their rate is variable and decided by the RTP. As long as the CSS does not give a new *target*, the RTP can optionally, either stay and wait at the last *target*, or extrapolate a next *target* upon the last ones, independently for each *variable*. This is our second way of getting round the linearity rule: it is a kind of "rubato" applied specifically to each *variable*.

Traps and oscillations are partly opposite to each other. Traps may occur if the strength is not an increasing function of the distance. Oscillations may occur, with too large a satisfaction step, if the strength is increasing with distance. Direction ambiguities may occur if two strength directions are possible with the same magnitude. We have no way of choosing.

Trace and data display are essential to any real time music system to enable the user to do twice the same thing. Display takes time. Trace takes memory. We choosed to trace in real time

and display the trace in deferred time. The user can trace, for each variable and each satisfaction step, the distance, the computed strength direction, magnitude and weight, the target, and the absolute (hard process) time from the preceding target.

Examples

The four examples on the graphic page show:

- (high left) a Hermite cubic, called "cherm" and defined in the text example by 6 points and derivatives (only 5 are visible). The thicker line in the middle is $C(t)$, the others are the equipotentials. The orthogonal lines show the way from different points to the curve.

- (high right) a polynomial called "cpoly" and defined by 3 coefficients.

- (low left) two concurrent circles with equal weight. One can see a trap in the middle between the two common points. As soon as the starting state is shifted or if a weight had been modified, the trap is left. One can also see an ambiguity point if the system is dropped on the common diameter.

- (low right) two separated circles with different weights. One can see - by observing the equipotentials - that the strength of one circle lowers down when the state is near the center of the other circle, thus far from it.

It is important to understand that the vision of these examples is a static vision of an essentially dynamic system. The curves may be parametered by the values coming from the RTP.

The POLE language is a LALR grammar built with YACC.

This example begins by a classical header, defining two default parameters, the step and

period of the satisfaction process. These data are decided, in real time, by the RTP.

After the header, some constraints are defined, especially a Hermite function defined by 6 points and derivatives, in a two dimension space.

The signs < > mean to one side or the other, = mean on the hyperplan.

```

library      mylib

parameter    step(2)[0] := 1, period := 10

procedure   Circle(2,3) Sinus(2,3) Logarithm(2,1)
Square(2,1)

strength    pull, push

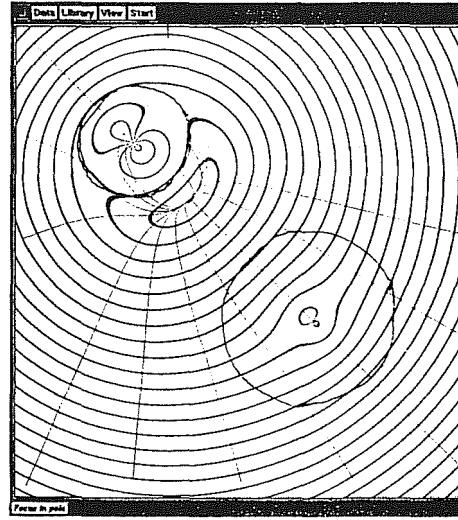
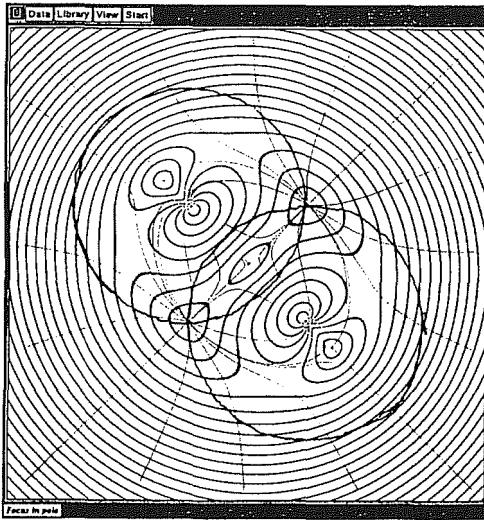
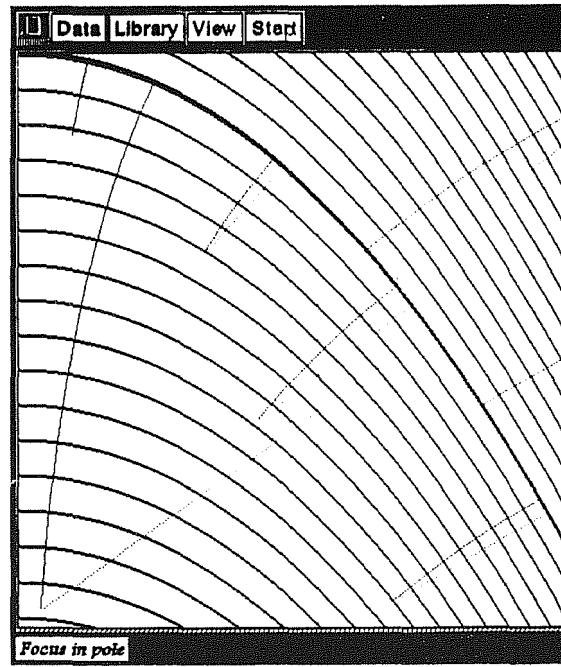
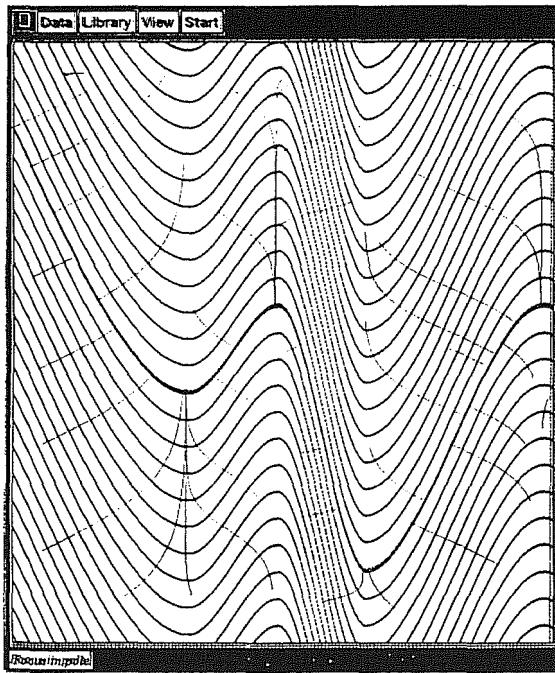
variable      x(3)[0] := 100, y := 200.3, z

-- Two parameters have been declared in the header,
therefor the index begins at 2
parameter    x0(3)[2] := 300, y0 := 300, my_parameter

constraint
-- in z[y], y is considered as a constant
cpoint :point x[100,200], y[20,my_parameter], z[y]
ccar :procedure Square(x,y;x0)
-- linear relationship x -2y + z + y0 = 0
clin :linear 1*x + -2*y + 1*z + y0
-- polynom y = 0 -2x + 0.5x^2
cpoly :polynom x, y, (0, -2, 0.5)
cherm :hermite x, y, (0,100,2), (200,400,0),
                (300,300,0), (400,600,0), (600,300,0), (800,800,1)
pole
pole0      : cpoint, =, pull, 1
pole1      : ccar,    <, push, my_parameter
pole2      : clin,    >, push, 1.2
pole3      : cherm,   =, pull, y0

character
car0 : 1, pole0, pole1
car1 : x0, pole1, pole3, pole2

```



Capitolo 7

STRUMENTI S/W E H/W PER LA COMPOSIZIONE E LA PERFORMANCE

MUSIC 5 MAC

Simone Bettini

C.S.C, Padova University

Via Gradenigo, 6a,

35100 Padova (Italy)

home telephone 0429/86230

E-Mail (internet) space@maya.dei.unipd.it

Music V and Music 5 Mac.

Music V is, from an historical point of view, a very important sound synthesis and music composition language.

Music V was the last of a series of Music languages carried out, starting from the end of fifties, by Max V. Mathews, researcher at Murray Hill Bell Laboratories [1]. It was the first language to allow the realisation of a virtual digital synthesiser.

It can be used to define, starting from their describing algorithms, the instrument of an "orchestra" to which pass a score to perform.

Music V, although it is now used mainly for didactic purposes, still stands out for its versatility and power and can be considered the first of a whole generation of music synthesis languages based on the same operation principles.

The first Music V compiler was implemented on the General Electric 650 computer of Bell Laboratories, then, with the increase of raw computational power, on smaller systems and

finally on personal computers.

In the beginning Music 5 Mac (M5Mac) was conceived as the Macintosh port of a PC Music V compiler written by Daniel Arfib, CNRS Marseille [2].

A preliminary straight port of the compiler resulted in a non Macintosh application: in particular it lacked a user-friendly graphical user interface allowing to simplify and speed up the scores writing. It indeed behaved as every other command line version of the package:

- the composer has to draw on paper a directed acyclic graph (DAG) representing the connections of the components of an instrument.
- the composer himself then translates this graph in the Music V language and completes the score adding the instructions relative to the motif execution. This sequence of instructions goes in a text file.
- such file is then passed to the Music V compiler which generates the sample file, eventually requiring long computational times.

To simplify this lengthy and

quite error prone procedure, at the same time speeding up instrument creation, it was then decided to add to this version of Music V a graphic user interface following the Macintosh Human Interface Guidelines [3]. Thus a complete visual development environment consisting in a text editor and a graphical editor was integrated in M5Mac beside the language compiler itself and other general

purpose tools.

Such an environment offers the composer powerful and flexible tools for score writing and allows to generate the sample files in background, in several output formats, and to listen directly to the result of computation.

The score development can thus be realised entirely in M5Mac following those steps:

- the DAG of the instrument is done at the computer using a dedicate graphic editor.
- then, instrument translation to the textual representation is executed automatically by a code generating procedure.
- the score can be completed in a text editing M5Mac window.
- the score is executed directly in M5Mac choosing the output format.

At the end of the samples generation it's possible to listen directly to the result.

Music 5 Mac features.

M5Mac manages two kinds of

data types: the scores(text) and instruments (DAGs).

Besides it can generate sample files in three formats: AIFF, Macintosh audio format ("snd" in "sfil" file) and 16 bit integers.

Two conversion routines between instruments and scores are available. The first allows to obtain the Music V language textual definition from its DAG, the second performs the opposite

operation: it allows to get an equivalent DAG for any instrument present in a score, pleasantly formatted on screen.

Several algorithms have been developed for those routines:

- tree analysis and code generation are done to translate DAGs to scores.
- text parsing, tree weighing and arranging are used to extrapolate a DAG relative to any instrument described in a score. This output can then be modified and reused.

This Music V version, among other things, allows reuse of already existent samples files as input data. 16 bit mono AIFF has been chosen as input format. Some conversion routines between the various formats generated by M5Mac have been specially written in order to ease the use of this feature.

M5Mac environment.

The global structure of the application has been developed following Apple's Human

Interface Guidelines:

- Standard File and Edit Menus, as well as application specific menus.
- Dialogs for opening and closing files and for other miscellaneous functions, such as confirmation dialogs to avoid accidental closing of files, with consequent loss of data.
- Full multitasking capability

The handling of the two different representations of instruments is carried out by two different types of windows (fig. 1).

For the score a normal text window, with usual cut, copy, paste possibility and an additional search and replace function is provided.

For the instrument DAGs a

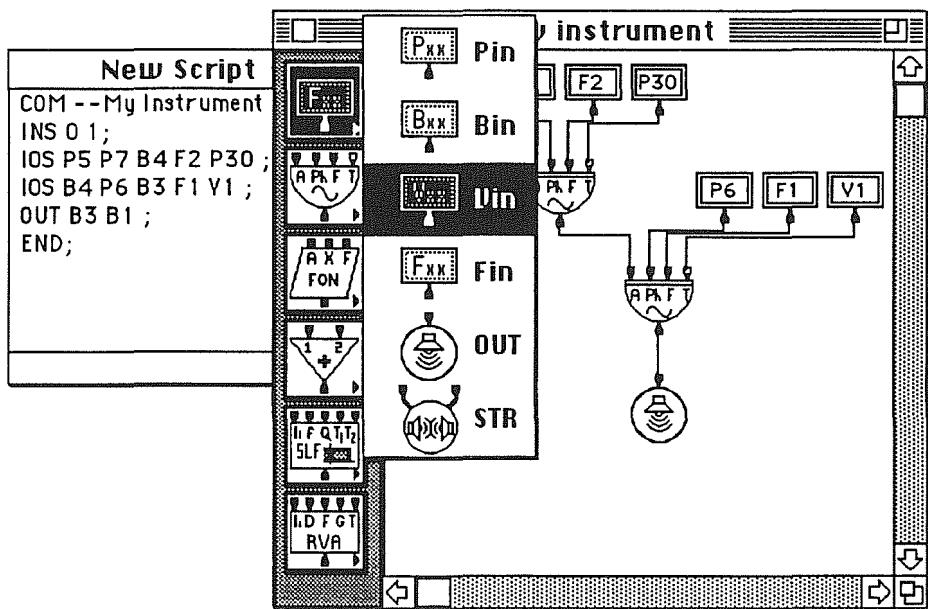


Fig.1: Music 5 Mac windows. The back window contains the textual definition of the instrument in the graphic one. On the left of graphic window there is the palette for the modules selection. The inputs menu is open.

(the user can perform other tasks while M5Mac compiles in the background).

- Multi-window environment with cut-copy-paste capability between windows.
- Possibility to print both scores and instruments diagrams through a consistent user interface.

dedicated window has been designed. It contains a palette with six pop-up menus from which it is possible to choose the modules for building instrument.

A dynamic data type in which coexist both a tree and a queue was conceived in order to manage the graphic representation of instruments.

The use of the graphic window and of the palette is very intuitive.

compilation, a movable modal dialog with a progress bar is shown (fig. 2). The user can stop

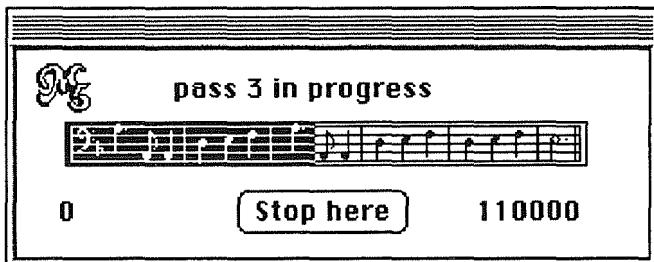


Fig. 2: The progress dialog

The modules can be joined in a easy and intuitive way simply clicking and holding the mouse button on an input [output] hot-spot, dragging to an output [input] hot-spot and releasing the button, thus forming arbitrarily complex pipelines. Eventual unacceptable connections (output-output, input-input) are not permitted.

The input pseudo-modules P, B, V, F icons show the value of the relative field. This value can be assigned using a command in the Edit menu.

The cursor shape changes according to the function performed (joining, selecting, dragging) to provide the user with good visual feedback.

It's possible to select one or more modules and copy, move or delete them. Connections between modules are copied along with the modules themselves.

Dragging the selection over the window borders causes the latter to scroll.

During score generation and

the current operation by hitting a button labeled "Stop" on this dialog. Doing this does not cause the loss of the already generated samples, which are saved in the output file, and then can be subsequently used.

Acknowledgement.

Roberto Avanzi gave some technical assistance during M5mac development and proofread this document.

References.

[1] Max V. Mathews and others: "*The Technology of computer music*", The M.I.T. Press, 1992.

[2] Daniel Arfib: "*Music V pour PC manual*", internal report, CNRS Marseille, 1990

[3] Apple Computer, Inc. "*Inside Macintosh*", volumes I-VI, Addison Wesley.

MELODIA: A PROGRAM FOR PERFORMANCE RULES TESTING, TEACHING, AND PIANO SCORES PERFORMING.

Roberto Bresin

C.S.C. - Centro di Sonologia Computazionale
via San Francesco 11 - 35121 Padova
phone: 049/8283757 - fax: 049/8283733
E-mail: rb@csc.unipd.it

Introduction

In the field of musical performance many researches have been made in the past years. One of the most interesting works is that developed by Sundberg and his collaborators at the KTH in Stockholm [1] [2]. They build a symbolic rules' system using the analysis-by-synthesis method. The purpose of these rules is to change a music score into an acceptable performance. The resulting deviations from the rules are additive because each note can be processed by several rules. These deviations made from each rule, are successively added to the parameters of that note. With this method expert musicians and teachers were asked to formulate some performing rules. These rules, from the suggestions of an expert musician, who judged whether the adding effect of the single rule was pleasant or not, were applied in sequence during

the performance. The performance is obtained through the sum of time, sound and timbre deviations due to the simultaneous application of numerous symbolic rules.

In the present work, I started from the results of Sundberg to build a program that is an experimental environment for teachers, students, and researchers to understand how performance rules can "modify" a nominal score.

MELODIA: the program

MELODIA is a program implemented at the C.S.C. laboratories in the field of the researches on automatic music performance [3] [4] [5]. In particular the program was first used to produce in output some data files suitable for the neural networks I built to perform in real time any music score. In a second

phase MELODIA was adapted to use it with a Yamaha Disklavier Grand Piano and to the needs of the professional pianist M° G.U. Battel. The program allows to perform via MIDI any score, previously written in a simple language, applying some symbolic rules. Many of these rules were chosen from those proposed by Sundberg and co-workers [1] [2], other rules are a modified version of Sundberg rules, and there are also completely new rules. In respect to the Sundberg rules system, MELODIA considers also rests, grace-notes, staccato, and legato. The application of rules occurs in an interactive way: each time the user can choose the rule to apply, and its weight.

When the choice of rules session is finished, the user is asked to listen to the results or to view them on the screen. The view on the screen shows both the time deviations graphic (in milliseconds) and the loudness deviations graphic (in decibel) due to the applications to the score of the chosen rules.

After this phase the user has the possibility to begin a new session with another score or to view and to listen to previously processed scores.

MELODIA, in the true sense of the word, can be used only on monophonic scores: in case of a polyphonic music, the program is

applied to the principal melody and then the other parts are synchronized with it.

MELODIA: performance rules

In the program there are 19 performance rules: most of them derive directly from the rules developed at the KTH and I remand to the related papers for a deep explanation of them. In the following there is a listing of these rules:

- 1 - Durational contrast
- 2 - Double duration
- 3 - High loud
- 4 - Faster uphill
- 5 - Melodic Charge
- 6 - Harmonic charge
- 7,8 - Chromatic charge (with and without rests)
- 9,10 - Leap tone duration (Sundberg and Bresin version)
- 11 - Leap articulation
- 12 - Articulation of repetition
- 13 - The shorter the shorter
- 14 - The shorter the softer
- 15 - Social duration
- 16 - Inegalles
- 17 - Accents
- 18 - Phrase
- 19 - Final retard

Some of these rules are a modified version of Sundberg's rules: the chromatic charge with rests considers also a context of notes with some rests between them; my version of the leap tone duration inverts the signs of the

equations to obtain the sensation of the physical movements of the hands of the pianist in a leap; the final retard rule has only the name of the same KTH's rule but acts in a different way.

Conclusions

Since the program allows a "micro" adjustment of all rules, it can be considered a valid tool to experiment performing rules. MELODIA can also be useful in the teaching of music performance, since it has the possibility to outline how the emphasis given to some notes in respect to others can change the performance of a particular composition. Furthermore MELODIA is a good tool to obtain score performances, which are "warmer" than those resulting from a performance without any micro-deviation or, like in some commercial programs, introducing some random deviations.

MELODIA gives also two output files, which contain the input patterns for two neural networks: one for time deviations, and the other for loudness deviations [5] [6].

The program runs on a personal computer IBM compatible, and with a Roland MPU-401 MIDI interface or compatible.

References

- [1] Sundberg J. et al. "*Performance Rules for Computer-Controlled Contemporary Keyboard Music*", Computer Music Journal, MIT Press, vol. 15, No. 2, pp. 49-55, Summer 1991
- [2] Friberg A. "*Generative Rules for Music Performance: A Formal Description of a Rule System*", Computer Music Journal, vol. 15, No. 2, pp. 56-71, Summer 1991
- [3] Bresin R., G. De Poli, A. Vidolin "*Un approccio connessionistico per il controllo dei parametri nell'esecuzione musicale*", Atti IX Colloquio di Informatica Musicale, pp. 88-102, Genova, 1991
- [4] Bresin R., De Poli G., Vidolin A. "*Symbolic and sub-symbolic rules system for real time score performance*", Proceedings of the 1992 ICMC, International Computer Music Association, San Francisco, 1992
- [5] Battel G.U., Bresin R., De Poli G., Vidolin A. "*Automatic performance of musical scores by mean of neural networks: evaluation with listening tests*", elsewhere in these proceedings, 1993
- [6] Rumelhart, D.E., McClelland, J.L. "*Parallel Distributed Processing*", vol. 1, Cambridge: MIT Press, chapter 8, 1988

MEDUSA: A POWERFUL MIDI PROCESSOR

Nicola Larosa, Claudio Rosati

IRIS s.r.l.

Parco La Selva, 151

03018 Paliano (FR), Italy

fax: +39 (775) 533343 - tel: +39 (775) 533441

E-mail: mc2842@mclink.it

MEDUSA is a MIDI device that works as a patcher, processor (filter, splitter, transposer, ...), and fader box.

The system core consists of an ASIC (Application Specific Integrated Circuit) single chip developed using semicustom technology (standard cells). It was designed within the European project OMI/DE-ARM (included in the program ESPRIT n. 6909).

The system, in its current prototype configuration, consists of two distinct parts: a rack module and a fader box, which are connected by a dedicated cable.

Internal structure

The system uses two or more ARM-60 RISC CPUs, MIDI serial interfaces, and fader box communication circuits.

The software architecture manages up to 32 MIDI inputs and 32 outputs. The multiprocessor hardware architecture allows one to assign the computational load to a CPU or distribute it among a num-

ber of CPUs, according to the required processing power. In the latter case, each CPU handles a subset of the MIDI inputs and/or outputs.

The first CPU (*the controller*) manages the fader box, the display controller, the LEDs, the buttons, the wheels, and the system parameters, sending them to the other CPUs (*the processors*) through a high speed parallel link.

The other CPUs work exclusively on MIDI data, fetching it from the inputs, processing and then sending it to the outputs.

Each processor performs two fundamental tasks: the *routing* and the *processing* of MIDI data.

Routing

Each processor can direct a message from its inputs to any output, independently and simultaneously with the traffic going on all the other connections.

This makes it possible to copy the messages from any in-

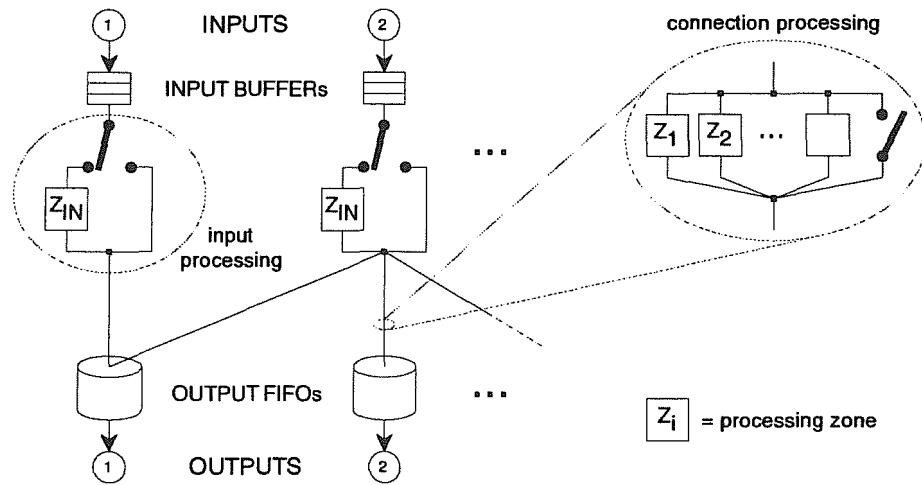


Figure 1: Processing flow diagram.

put to any number of outputs, and also to merge the messages coming from any number of inputs to any output.

The routing of those messages which have a predefined length is carried out in four steps (see fig. 1):

1. the message is stored in an input buffer regenerating the status byte, when using "running status";
2. the message may undergo processing by the zone (see below) associated with that input (input processing);
3. for each connection of that input, the message may undergo processing by having up to 128 zones associated with that connection (connection processing);

4. the message is inserted into the FIFO of all those outputs which are connected to that input, thus achieving the merge.

System Exclusive messages require a different treatment, in that they can have arbitrary length and cannot be interrupted by other messages, apart from System Realtime.

When a few inputs are being merged, the arrival of a SysEx message on one input necessarily delays the processing of the messages on the other inputs, until the end of the SysEx message. If the latter is considerably long (even 100 ÷ 150 Kbytes), the delay can reach a few dozen seconds.

To avoid, as much as possible, any data loss, the SysEx messages are inserted into dedicated buffers (not shown in

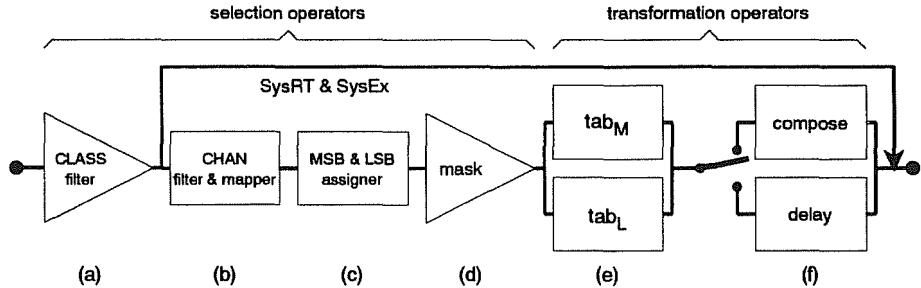


Figure 2: Zone flow diagram.

fig. 1), which are dynamically assigned.

When merging n inputs, the average flow of each input must exploit no more than

$$\frac{3125 \text{ byte/s}}{n},$$

where 3125 byte/s is the full MIDI bandwidth. Heavier flow peaks are allowed within a time frame dependent on the output FIFO's length.

As an example, for a 256 byte FIFO, the merge of 4 full bandwidth inputs is allowed inside a time frame of about 20 ms.

Processing

The processing of messages is based on the concept of *zone*.

The zone can be pictured with a set of *selection operators* and a set of *transformation operators*.

The message to be "transformed" is selected by three *filters* (see fig. 2):

- (a) selectively allows the passage of those messages be-

longing to one or more *classes*,

- (b) selectively allows the passage of those messages belonging to one or more MIDI *channels*,
- (d) selectively allows the passage of those messages whose values fall inside a predefined set.

The (b) operator, besides the function of channel filter, can "bump" a message from any channel to any other.

The selection (d) operator uses two filtering masks of 128 bits each, whose meaning is defined by the assignment (c) operator (see fig. 3) according to the message class.

The thus selected messages go through two transformation operators:

- (e) independent remapping of the values assigned by the (c) operator, using two tables, as follows:

$$\text{MSB} = \text{tab}_M[\text{MSB}_{\text{old}} + \text{offset}_M],$$

$$\text{LSB} = \text{tab}_L[\text{LSB}_{\text{old}} + \text{offset}_L];$$

message	MSB	LSB
Note On	key	velocity
Note Off	key	velocity
Poly Press	key	pressure
Chan Press	pressure	0
Prog Change	program	0
Ctrl Change	controller	value
Pitch Bend	value _{MSB}	value _{LSB}
MTC	data	0
Song Position	value _{MSB}	value _{LSB}
Song Select	song	0
F4+F6	0	0

Figure 3: MSB & LSB assignment.

- (f) delay or *composition* of the message, global or independent for each one of the 128 values of MSB or LSB, which are selectable.

The composition process sends byte strings in place of the triggering message. The strings are user definable and stored in dedicated buffers.

From the message a few parameters can be derived, and these can be used in the composition. Among them, there are the MIDI channel, the MSB and LSB values, and their complements.

In the transformation (f) operator it is possible to set the delay time (from 0 ms to 10 s), or else set the index of the composition buffer to be triggered by the messages, independently of their values.

As an alternative, it is possible to choose a table (addressed by MSB or LSB) where the delay times and/or the

indexes of the composition buffers are stored.

For example, selecting the Note messages, one could assign greater delay times to notes with greater key values. At the same time, a few unused notes could be composed in System Realtime Start and Stop messages.

Fader Box

By means of sliders, buttons and a joystick, the fader box may generate any MIDI message or group of messages, using specific buffers which are definable by the user.

More simply, a group of modes is also available, instantly recallable, where the controls send the most common messages by default.

Conclusion

MEDUSA was designed focusing on two main aspects:

1. power and flexibility;
2. ease of use.

The system includes many of the features of MIDI patch-bays, master keyboards and processors on the market, plus a few not currently available.

This feat has been achieved by means of a configurable architecture, caring software, and user friendly devices such as a large LCD, wheels and joystick.

PWCONSTRAINTS

Mikael Laurson

Tietokonemusiikistudio

Sibelius Akatemia

PL 86

00251 Helsinki (Finland)

E-mail Mikael@next1.siba.fi

Abstract: *PWConstraints* is a rule-based constraint language working under PatchWork (PW) (Duthen, Laurson, Rueda). It can be used to fill pitch information of musical textures. Currently the system can write classical counterpoint, solve problems using musical set-theory to generate rows, matrixes, chains, harmonic structures, etc.

PWConstraints provides a backtrack search engine, a genetic search engine and a traditional musical score representation.

If the user has a specific musical problem she/he can write rules as Common Lisp functions, without having to worry about backtracking, creating musical structures from scratch, other rules, etc.

Rules are modular in the sense that they can be in any order and

that you can insert or take away a rule without affecting the rest of the system.

The user specifies the time structure of the problem in a PW rhythm editor module as a traditional score either by hand or by algorithm. During search the score is providing the rules information about musical context of each note.

A part of a problem can be constrained in advance and the task of the search is to fill the unconstrained parts. This makes the system much easier and natural to control because you do not have to invent rules for those parts that are constrained in advance. Examples where this kind of constraining might be useful are cadences in classical counterpoint, cantus firmus techniques, etc.

Because the user works directly with a traditional score, making

those constraints is as straightforward as writing notes on paper.

The result of the calculation is written directly to the score, so reading, listening and printing the result is easy.

For the evaluation of the result the user can use analysis tools that are available in PatchWork, like the set-theoretical analysis

package *PWMacSet* (Castren, Laurson).

PWConstraints can also be used as an analysis tool where a score is given as an argument to the rule system. In this case all pitches of the given score are constrained and the system loops through the score. If a rule fails, a message is printed giving information about the rule and the exact location in the score.

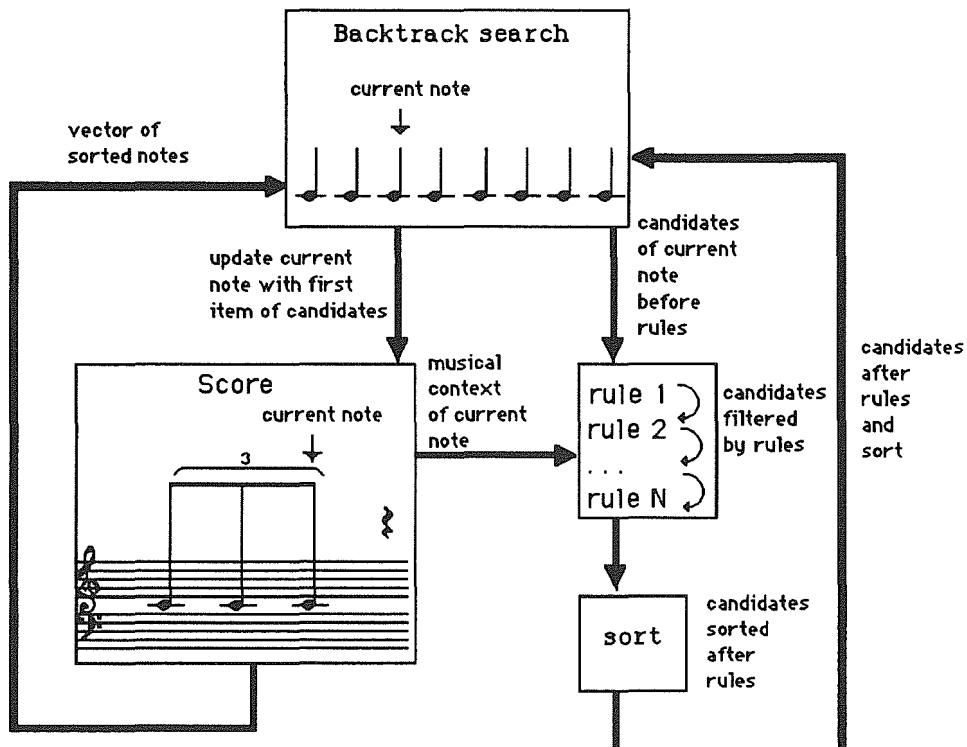


Figure 1. Overview of the system

PWConstraints consists of three parts: score, search and rules (figure 1).

The search space is defined at the preparation phase by taking all the ordered note objects from a score and assigning to each of them a range of possible pitches.

For example the task might be to write a diatonic five note melody for a soprano voice with a range of 1.5 octaves - in midi values from 60 to 77).

This means that we are assigning to each of the five notes a list of midi values - in this example the list would be (60 62 64 65 67 69 71 72 74 75 77). All solutions for this problem can be found by taking all possible five-note paths by selecting one midi value from each range list, more formally by taking the Cartesian product of all range lists (or candidates).

The search engine is a vector of ordered note objects. The search can proceed in both directions. A current note is selected from the beginning or the end of the vector. All current rules are called with the same argument list that contains information of the current note, its list of candidates and its surrounding in the score, etc.

Each rule acts as a chained filter, where the list of candidates is continuously filtered. After all rules have been applied this filtered list represents all possible midi values that a note object can have in this specific musical context. This list is sorted according to a user definable function. The first item of the filtered list is written as the pitch value of the note and the rest of the list is stored in the note object. The new pitch value changes the state of the score and next note objects can refer to this change when the musical status is been analysed later in the search. The rest of the list acts as a kind of buffer if later the search has to backtrack. If backtracking occurs the first item of the list is popped and written as a new pitch value of the current note without having to run the rules again. The status of the score has changed and the search engine can go on.

If the filtered list is empty the search has to backtrack and go to the previous note object (default behaviour). Otherwise the search engine can select the next note object as current and continue the search. A rule can control the backtrack mechanism by jumping directly to any previous

location in the score. For instance if a melodic rule fails it can jump directly to the note that caused the failure, instead of having to backtrack to that note step by step.

Figure 2 shows a short example using set-theory to create a hexachord chain with three embedded tetrachords.

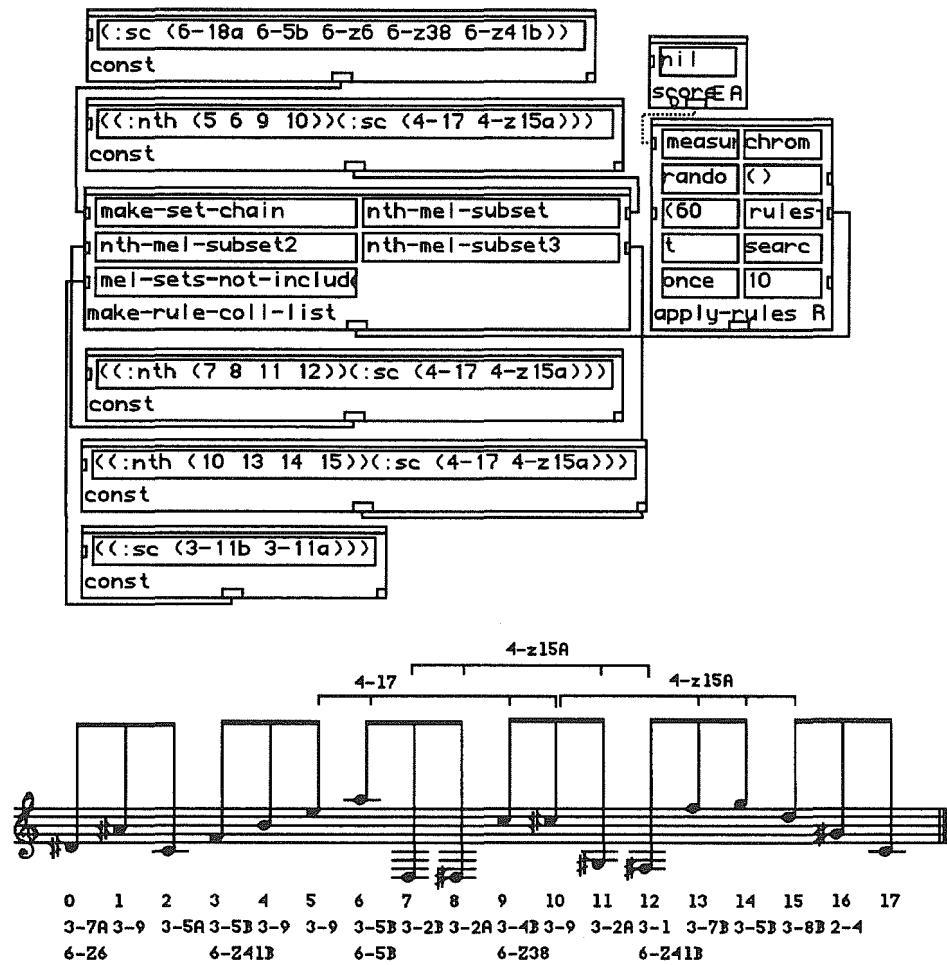


Figure 2. A set-theoretical example. Above a PW patch and below the result.

SoundLib 2.0

Una libreria di classi C++ per l'elaborazione di segnali audio campionati

Andrea Provaglio

C.C.A., Università di Padova
via San Francesco, 11
I-35121 Padova (Italy)
fax +39 49 8283733
E-mail music01@unipad.unipd.it

Abstract

This paper describes the main design strategies adopted in the development of a C++ class library for the processing of sampled audio signals, namely SoundLib 2.0. The design intentionally bear resemblances with the standard library *iostream*, but there are no inheritance relationships between the two libraries. A *Parent-Child* mechanism among instances of some classes of SoundLib was used to combine different abstractions.

Operator overloading and friend functions was largely used to achieve a simple syntax in the use of the classes.

Introduzione

SoundLib 2.0 è una libreria di classi C++ rivolta specificatamente allo sviluppo di software per l'elaborazione di

segnali audio campionati. Per quanto possibile, il progetto non è stato legato a una data piattaforma hardware e/o software, e la libreria esiste al momento in versione per DOS e per Windows. In fase di progetto particolare attenzione è stata posta nel creare delle astrazioni modulari e fortemente riutilizzabili, e nel fornire una sintassi per l'impiego delle stesse che fosse il più semplice possibile. Nel design di SoundLib ci si inoltre è ispirati alla libreria standard *iostream*, per l'evidente analogia di alcune funzionalità, per l'eleganza della sintassi e per la comprovata robustezza della architettura; d'altro canto, essendo evidenti dei casi in cui l'analogia mancasse totalmente, o non ci fosse compatibilità nel design, non vi sono tra le due librerie relazioni di carattere ereditario; la somiglianza con *iostream* è prevalentemente a livello sintattico, come nell'uso

degli operatori di inserimento ed estrazione.

Applicazioni

Lo sviluppo della libreria è stato guidato dalla analisi dei classici problemi che si incontrano nella realizzazione di applicazioni per il trattamento di segnali campionati; le operazioni per le quali SoundLib è stata progettata sono state divise in categorie funzionali, di seguito illustrate:

Trasferimento di campioni tra diverse aree del sistema

Si tratta di operazioni a basso livello, quali il colloquio tra driver di dispositivi di conversione AD/DA e buffer di memoria, operazioni di scrittura e lettura di buffer di campioni da supporti di massa e scambio di campioni tra buffer di memoria.

Operazioni di marcatura e selezione

Sono le operazioni con le quali è possibile indicare particolari punti o aree del segnale.

L'implementazione di questo tipo di funzionalità, apparentemente semplice, ha in realtà avuto notevole influenza sul design della libreria, portando alla architettura *Parent-Child* più avanti descritta. Le operazioni di marcatura permettono di indicare un particolare punto del segnale. Questi marker possono essere

legati a un istante reattivo all'inizio del segnale, o a un evento nel contenuto del segnale; un esempio del primo caso sono i marker utilizzati nei missaggi, mentre un esempio del secondo caso sono i marker utilizzati nella segmentazione del parlato. La differenza tra i due tipi di marker consiste nel fatto che i secondi vengono automaticamente spostati ad un diverso istante in seguito ad operazioni di editing sul segnale, verificate a monte del marker, che abbiano modificato la durata del segnale.

Le operazioni di selezione permettono di indicare su quale area del segnale si vuole operare, per esempio la zona da visualizzare sullo schermo, da filtrare o da analizzare. Sono stati identificati due tipi di selezione: temporale e spaziale; una selezione temporale specifica l'area del segnale in termini di intervallo di tempo, mentre una selezione spaziale indica su quali canali operare. Questi due tipi di selezione possono essere combinati, ed hanno le stesse proprietà dei marker sopra descritti.

Elaborazione di campioni

Si intendono tutte le operazioni che modificano il segnale, le più semplici delle quali sono le operazioni di *cut and paste*. Altre operazioni di elaborazione sono i

vari tipi di missaggi, che tra l'altro estendono le operazioni di inserimento e sovrascrittura tipiche del *cut and paste*; queste comprendono il cross-fade, fade-in e fade-out. Infine vi sono i filtraggi, intesi qui in senso vasto, in quanto si intendono operazioni di equalizzazione, riverberazione, time stretching e così via; sono tutte le operazioni algoritmiche che alterano il contenuto spettrale del segnale.

Analisi del segnale

Sono quelle operazioni che restituiscono informazioni sul contenuto del segnale, dalla semplice durata, o picco massimo, fino al contenuto spettrale, all'andamento del pitch e così via.

Generazione algoritmica di segnale

I moduli di generazione algoritmica includono i generatori di rumore bianco, di silenzio e i generatori di forma d'onda.

Descrizione

Concettualmente, il centro della libreria è una classe astratta denominata *Sound*, da cui discendono le due classi astratte *ISound* e *OSound*, che definiscono le modalità di inserimento ed estrazione di campioni con una sintassi analoga a quella di *iostream*, nonché

funzionalità di interrogazione dello stato dell'oggetto.

Dalla classe *ISound* discende tutta la famiglia dei generatori algoritmici di segnale, di cui la libreria implementa attualmente solo i più semplici. Una applicazione dei discendenti di *ISound*, attualmente in fase di valutazione, è la realizzazione di classi che supportino la generazione di campioni tramite partitura in un linguaggio per la sintesi del suono (*csound*, *Music5*).

Da *ISound* e *OSound* discende, in analogia con *iostream*, la classe astratta *IOSound* da cui a loro volta discendono la maggior parte delle classi non astratte della libreria, tra cui *SoundChannel*, *SoundSelection* e *SoundMarker*; queste astrazioni forniscono le funzionalità necessarie per le operazioni di marcatura e selezione sopra descritte, e sono state determinanti nella scelta della architettura *Parent-Child* tra le istanze di queste classi (si noti che la relazione *Parent-Child* non va qui intesa come di tipo ereditario). Definiamo *child* un oggetto di una classe discendente di *Sound* quando questo gestisce indirettamente campioni, forniti da un oggetto *parent* specificato nella creazione del *child*. Il *parent* tiene traccia di tutti i propri *child*, in modo da poter notificare eventuali variazioni nel segnale

che li possano interessare. Le classi che gestiscono direttamente i campioni quindi possono essere solo *parent*, *SoundChannel* e *SoundSelection* possono essere sia *parent* che *child*, mentre *SoundMarker* può solo essere solo un *child*. L'unione tra le relazioni ereditarie della maggior parte delle classi e la architettura *Parent-Child* sopra descritta permette di combinare in maniera estremamente semplice, ma flessibile e potente, le diverse astrazioni messe a disposizione da SoundLib (si veda l'esempio 1).

Si è voluta mantenere la stessa sintassi e lo stesso approccio concettuale anche nella gestione dei dispositivi di conversione AD/DA, dei soundfile e dei buffer di memoria, facendo derivare le relative astrazioni a livello più alto dalla famiglia di *Sound*. Questo, unitamente ad una particolare cura nella implementazione del meccanismo di dialogo tra istanze delle classi, permette di utilizzare

queste astrazioni in maniera estremamente comoda.

Comunque, classi a livello più basso sono state sviluppate per gestire il formato dei soundfile e l'utilizzo delle risorse di sistema.

Opportuni manipolatori e *friend function* della famiglia di *Sound* permettono, con una sintassi simile ad *iostream*, di impostare i parametri per la acquisizione e riproduzione del segnale, di effettuare filtri, e così via.

Conclusioni

SoundLib è già stata impiegata con successo nello sviluppo di applicazioni complete e funzionanti, semplificandone notevolmente la realizzazione. L'approccio scelto per il design si è rivelato particolarmente solido e efficace.

Ringraziamenti

Si desidera ringraziare Nicola Bernardini per il brainstorming, Graziano Tisato e Roberto Cavazzana per la collaborazione.

Esempio 1

```
// Questo esempio apre un soundfile stereo esistente,  
// ne azzerà i campioni da 1000 a 2000 del canale sinistro  
// e chiude il file ("silence" è fornito dalla libreria).  
void main(void)  
{  
    DiskSound aStereoSound( "DEMO.WAV" );  
    SoundChannel leftChannel( aStereoSound, CHAN_LEFT );  
    SoundSelection noisy( leftChannel, 1000, 2000 );  
    silence >> noisy;  
}
```

"HIPPOPOTAMUS"

Un sistema di performance interattivo

Giovanni Ramello

(A.M.E.T. - Conservatorio G. Verdi di Torino)

via Torricelli, 5 - 10128 Torino

fax ++39.11.507141

Abstract

This work has been produced at the "Banff Centre for the Arts" for an art exhibition's musical installation in an ex-zoo.

The original idea was to create a dynamic musical process with chosen degrees of player's interaction with the machines.

The controllers used to input new parameters and musical events are two: a standard keyboard "re-interpreted" by Max (the Ircam object oriented language running on a Macintosh platform), to answer in a different way to the gestures of the traditional instrumental technics; the second interface is the monitor, with his own virtual sliders, keys and buttons, mouse controlled.

The system, essentially, is controlled by two different algorithms written using Max,

to realize the two movement of the piece, "The hippopotamus steps", with an onomatopeitic flavour, and "The jungle sounds", the most extensively interactif. The keyword of "Hippopotamus" is the interplay between the human being and the machine.

Introduzione

Questo lavoro, realizzato nell'estate 1992 presso il Banff Centre for the Arts, è nato come audio-installazione per i locali di un ex zoo ed è stato successivamente modificato per esibizioni concertistiche.

L'intenzione iniziale era di creare un flusso sonoro in tempo reale, continuamente in divenire e gli strumenti della composizione algoritmica si sono rivelati particolarmente adatti a tale scopo.

Ripartendo dalla definizione reichiana della musica come "processo graduale" [1], abbiamo cercato, dunque, di

costruire un processo dinamico piuttosto che una partitura per computer, ma nel contempo abbiamo sentito la necessita' di inserirvi un contributo umano attivo, mediante alcuni gradi prescelti di interazione di un esecutore / utente con il meccanismo stesso. L'utilizzo, poi, di variabili casuali all'interno degli algoritmi e di una partitura "aperta", ricollega questo lavoro, al meno in termini di ispirazione, ai molteplici studi ed esperimenti sull'indeterminazione nella musica, realizzati dalle scuole della musica stocastica, della musica intuitiva ed improvvisata [2] con un tributo particolare agli insegnamenti di John Cage sull'alea in musica.

La direttiva principale seguita e' stata quella di disegnare un sistema che combina l'umano e l'artificiale secondo una logica di non preminenza, in una sorta di gioco strategico in cui uomo e macchina dialogano ed agiscono quali partners, interdipendenti tra loro. La chiave di volta del processo e' l'"interplay": il computer genera alcuni eventi musicali, l'esecutore puo' modificarli e aggiungerne nuovi, la macchina puo' a sua volta alterarli o semplicemente "impararli" e riproporli piu' o meno variati e cosi' via. Il prodotto musicale

che ne scaturisce porta percio' il segno delle scelte combinate di esecutore e computer, il quale, ovviamente, si muove a sua volta entro i limiti prestabiliti dal compositore/programmatore.

Dal punto di vista esecutivo, il sistema e' stato realizzato in modo tale da liberare il "performer" dalla tradizionale corrispondenza gesto/suono tramite la "reinterpretazione" di una normale tastiera elettronica e la realizzazione di uno specifico controller virtuale su monitor, accessibile via mouse.

Materiali

"Hippopotamus" utilizza un expander Yamaha TG77 per la sintesi sonora, una tastiera Yamaha SY55 come master keyboard, 2 DSP Yamaha SPX 900, un mixer 8 su 4 e relativo impianto di amplificazione e diffusione, un Macintosh con porte MIDI e almeno 2 Megabytes di memoria, e software scritto con il linguaggio object oriented Max dell'IRCAM.

La configurazione del sistema MIDI e' la seguente: SY55 out --> in Macintosh out --> in TG77.

La parte audio invece e' stata di volta in volta adattata alle esigenze ed alle disponibilita' contingenti, sebbene la soluzione ottimale sia quella quadrifonica (doppia stereofonia).

I controllers a disposizione dell'esecutore sono la master keyboard, con due possibili

configurazioni e il monitor del computer (anch'esso con due algoritmi alternativi), nel quale l'utente, attraverso il mouse, puo' pilotare il processo agendo su sliders e pulsanti virtuali.

Metodi

Il sistema "Hippopotamus" e' formato da due algoritmi; il primo, il piu' semplice, trasforma modulation wheel e data slider del SY55 in veri e propri strumenti musicali, attraverso cui controllare, contemporaneamente i codici nota e la velocity di 2 diversi timbri del TG 77, dando vita al primo movimento "I passi dell'ippopotamo" di sapore vagamente onomatopeico.

Il secondo algoritmo, utilizzato per il movimento successivo, "I suoni della giungla", contiene tre generatori di eventi casuali sui quali l'esecutore puo' intervenire e che, a loro volta, si comportano anche da elaboratori dei dati immessi dall'utente tramite i controllers.

Due di questi generatori sono sostanzialmente uguali tra loro, presentano un livello d'interattività piu' limitata e servono a creare un doppio flusso di eventi sonori, che rappresenta, anche timbricamente, una "foresta pluviale". Il terzo, invece, genera gli interventi degli "abitanti" della selva e, pertanto, produce selettivamente le apparizioni musicali

di quattordici differenti timbri.

Per ottenere questa alternanza di canali MIDI, senza incontrare problemi di noteoff, soprattutto in seguito a figure ritmiche tra timbri diversi particolarmente rapide (causa in sede empirica di notevoli difficolta'), abbiamo realizzato un oggetto[3] chiamato "filtro" che accende e spegne 14 sub-generatori (uno per ciascun timbro) permettendo cosi' di mantenere stabili i flussi di codici MIDI, senza "channel changes".

L'interazione dell'esecutore con i due primi generatori e' vincolata all'on/off del processo e all'eventuale inserimento di note che il computer elabora.

Piu' capillare e' invece la presenza umana nel terzo, in cui l'utente puo' come al solito inserire eventi nota, accendere e spegnere il processo nel suo complesso, ma anche nelle sue singole parti cosi' da avere un ruolo determinante nell'orchestrazione, nella densita' sonora e nella presenza delle pause in quella che potremmo definire la parte solista. Inoltre, attraverso uno slider virtuale si puo' determinare la velocita' di elaborazione e di risposta del generatore stesso agli inputs "live".

Le variabili casuali utilizzate sono quelle interne al linguaggio (istruzioni del tipo

"random"). Per evitare connotazioni troppo marcate sui risultati musicali, piu' istruzioni "random" sono state collegate, rispettivamente, in parallelo o in serie, a seconda della necessita' di delimitare o meno l'intervallo di estrazione.

Considerazioni

La frequentazione dei sistemi interattivi apre interessanti prospettive creative a scapito, pero', di nuove problematiche.

Questo tipo di produzioni porta, in effetti, ad una ridiscussione della posizione del compositore, tradizionalmente inteso, nel processo creativo.

Da una parte questi viene a perdere quel potere di determinazione pressoché totale sul prodotto musicale che gli procura la partitura scritta, ma ne acquista in termini di quantita' di parametri di scelta e di controllo. D'altronde, se vale la definizione di musica come "rumore messo in forma" [4] secondo sintassi mutevoli, il computer (grazie alla flessibilita' di strumenti quali Max e l'ambiente MIDI) offre liberta' pressoché illimitata nell'ordinare il dominio delle frequenze udibili.

La stessa apertura e', paralleamente, lasciata al performer, che si trova in una posizione meno coercitiva e piu' reattiva rispetto a quanto avviene con gli strumenti tradizionali (ov-

-viamente tutto cio' non implica dei giudizi assoluti di valore).

A questo proposito, la sperimentazione di "Hippopotamus" da parte di artisti, anche non musicisti, ha testimoniato una soddisfazione quasi liberatoria (soprattutto da parte degli strumentisti) nell'uso del sistema per produrre e manipolare il materiale sonoro, senza i patemi e le inibizioni che si verificano nella tradizionale pratica musicale.

Un'attenta frequentazione del sistema, infine, ha permesso di scoprirne idiosincrasie e qualita', favorendo applicazioni a differenti contesti e sviluppi compositivi ed esecutivi non considerati in fase progettuale.

Bibliografia

- [1] S. Reich, "Écrits et entretiens sur la musique", C. Bourgois Ed., pp.49-51, 1981.
- [2] C. Roads, "Introduction" in "Composers and the computer", C. Roads ed., pp.XI-XXI, W. Kaufman Inc., 1985.
- [3] D. Zicarelli, "Writing External Objects for Max", Opcode Systems, 1991.
- [4] J. Attali, "Rumori. Un'economia politica della musica", p.38, Mazzotta, 1978.

LE TWIN TOWERS UN DISPOSITIVO PER ESECUZIONI INTERATTIVE DI COMPUTER MUSIC

L.Tarabella, G.Bertini, M.Romboli

Computer Music Department of CNUCE/C.N.R.

Via S.Maria 36, I-56126 Pisa

tel. +39-50-593276 fax +39-50-904052

e.mail:music5 @ icnucevm.cnuce.cnr.it

Abstract

The Twin Towers here described is a device for controlling interactive computer music which detects information from the movements of hands, and however does not require any kind of physical connection with them.

It consists of two sets of four sensing devices which create two zones of the space (the vertical edges of two square-based parallelepipedon, or *towers*) where an object can be detected in terms of distance and front and side rotations, with respect to a reference frame.

Each sensing device actually consists of an infrared-based ranging system composed by a transmitting diode and a receiving diode.

Using the values of the four heights of each tower, it is possible to compute the heights and the rotations of the hands.

The whole system so implements a sort of two aerial three dimensional joy sticks.

Introduzione.

In questi ultimi anni sono stati realizzati e/o utilizzati molti dispositivi in grado di rilevare informazioni dal movimento delle mani (e delle dita) per controllare sistemi interattivi di computer music, come ad esempio il Data Glove, The Hands etc. [1] [2]

Le Twin Towers qui descritte costituiscono un dispositivo che rilevando il movimento delle mani rientra in questa categoria e che tuttavia non richiede di avere alcun contatto fisico con parti elettromeccaniche.

Le Twin Towers sono state realizzate in collaborazione con il Reparto Elaborazione Segnali ed Immagini dell'IEI/CNR, Pisa, come evoluzione dell'Infrared Pyramid realizzata presso l'ARTS (Advanced Robotics Technology and Systems) Lab, Pisa. [3]

Le Twin Towers.

Il dispositivo consiste di due insiemi di quattro sensori che creano due zone dello spazio do-

ve un oggetto può essere rilevato in termini di distanza e di orientamento rispetto ad un sistema di riferimento. Ciascuno dei due insiemi consiste di quattro sensori posti ai vertici di un quadrato di 5 centimetri circa di lato ed attivi dal basso verso l'alto; i raggi sono perciò idealmente posti sugli spigoli di

due torri quadrangolari poste a loro volta a 30 centimetri circa di distanza l'uno dall'altro.

Con i valori delle altezze raggi di ciascuna torre vengono calcolate altezza e rotazioni di ciascuna delle due mani: il sistema realizza così una sorta di doppio joy stick aereo.

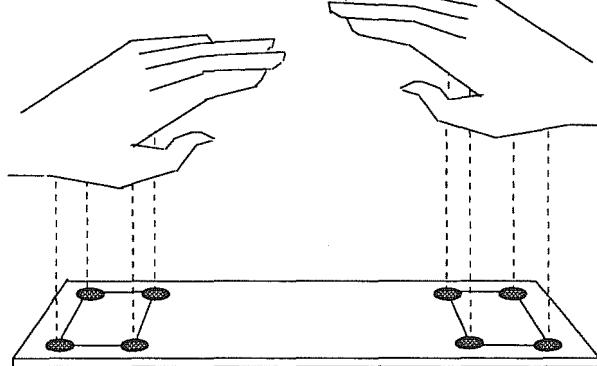


fig.1 - Le Twin Towers

Realizzazione.

Ciascun sensore è costituito da due fotodiodi di cui il primo trasmittente ed il secondo ricevente del raggio riflesso dall'ostacolo incontrato.

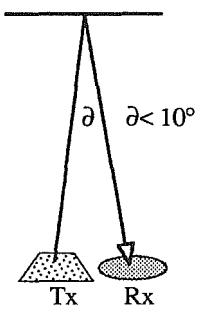


Fig.2 - Fotodiodi

L'elettronica di controllo è composta da un sistema di trasmissione multiplexato che pilota gli 8 diodi trasmittenti ad infrarossi, e da 8 sistemi di ricezione. Il sistema di trasmissione è composto da un generatore di impulsi e da un amplificatore di potenza. La forma del segnale di controllo è un'onda quadra a ≈ 130 Hz con lo 0.01% di duty cycle. [4][5]

Il segnale fornito da ciascun diodo di ricezione, viene trattato con un filtro passa-alto che elimina i disturbi dovuti alla luce di sorgenti artificiali e mandato ad un amplificatore driver che fornisce così una tensione proporzionale alla distanza tra

l'oggetto rilevato e la sorgente ad

infrarossi stessa.

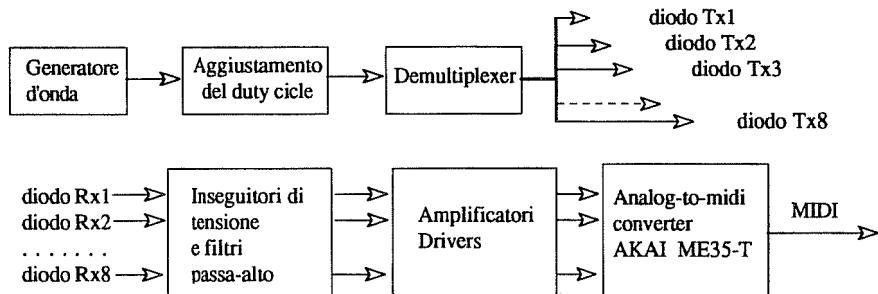


Fig.3 - Schema a blocchi dell'elettronica di controllo

La circuiteria di controllo delle Twin Towers manda i valori analogici degli 8 trasduttori agli 8 ingressi di un Analog-to-MIDI trigger (AKAI ME35T); un insieme di 8 valori viene spedito ogni decimo di secondo, e l'Analog-to-MIDI trigger è programmato per generare messaggi

MIDI NoteOn dove il Key-Number individua il numero di trasduttore (1÷8) ed il Velocity riporta il valore dell'altezza.

La risoluzione in altezza rilevata da ciascun sensore è di circa 2 millimetri e l'intervallo di azione significativo è di circa 25 centimetri.

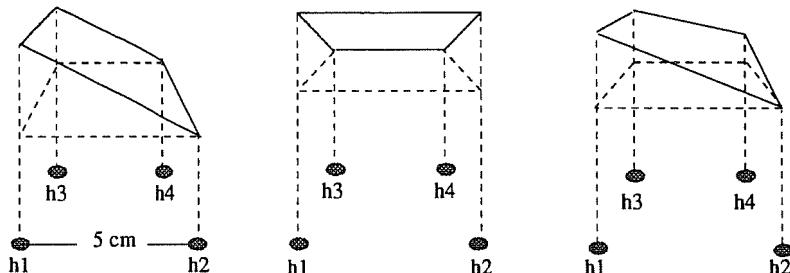


Fig.4 - Tre differenti possibili situazioni di una singola torre

I Messaggi NoteOn generati dall'Analog-to-MIDI trigger vengono utilizzati da un programma attivo su un personal computer che in base alle 4 altezze di ciascuna torre calcola altezza media e rotazioni laterale e frontale delle due mani:

$$\text{Altezza media} = \frac{h_1 + h_2 + h_3 + h_4}{k}$$

$$\text{Rotaz. Later.} = \text{atan} \left(\frac{h_1 - h_2 + h_3 - h_4}{k} \right)$$

$$\text{Rotaz. Front.} = \text{atan} \left(\frac{h_1 - h_3 + h_2 - h_4}{k} \right)$$

Il programma genera poi una serie di messaggi MIDI Control Change in base alla seguente tabella di riferimento:

Bn 5A vv	altezza m.d. (mano destra)
Bn 5B vv	rotazione laterale m.d.
Bn 5C vv	rotazione frontale m.d
Bn 5D vv	altezza m.s.
Bn 5E vv	rotazione laterale m.s.
Bn 5F vv	rotazione frontale m.s

Conclusione.

Quella dell'utilizzo di un analog-to-MIDI trigger ed un personal computer deve essere considerata una soluzione transitoria: la forma finale dell'apparato vedrà infatti incorporato anche un modulo di generazione di messaggi MIDI.

Per le sue caratteristiche questo dispositivo può essere inserito nella categoria dei controller MIDI in quanto i messaggi generati possono essere utilizzati per controllare in real-time qualunque apparato di generazione sonora dello standard MIDI. Va tuttavia sottolineato che lo scopo principale per cui le Twin Towers sono state progettate e realizzate è quello di controllare composizioni musicali interattive realizzate con l'uso di sistemi e linguaggi tipo MAX, PascalMusic ed altri.[6] [7] [8]

Ringraziamenti.

Si ringrazia l'Ing.V.Genovese dell'ARTS Lab di Pisa per averci dato il supporto tecnico relativo

alla sensoristica tipica del settore della robotica.

Bibliografia

- [1] M.Waisvicz: "*The hands, a set of remote midi controller*" ICMC 84 Proceedings, pp. 313-318,1984
- [2] R.B.Knapp, H.S. Luster: "*A bioelectric Controller for Computer Music Application* - CMJ Vol.14, n.1, pp. 42-47,1990.
- [3] V.Genovese et alii: "*InfraredBased MIDI Event Generator*", Proceedings of the International Workshop on Man-Machine Interaction in Live Performance, Servizi Tecnografico, Area di Ricerca, Pisa, pp. 1-8,1991.
- [4] M. Idesawa: "*Optical Range Finding Methods for Robotics*" Proceedings of the IEEE International Conference on Robotics and Automation, pp. 207-213, 1986.
- [5] B. Espiau and J.Y. Castros: "*Use of Optical Reflectance Sensors in Robotics Applications*", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-10, No.12, pp. 903-912, 1980.
- [6] L.Tarabella: "Informatica e Musica", Jackson Libri, 1992.
- [7] M. Puckette, "*Combining Events and Signal in the MAX Graphical Programming Environment*", Computer Music Journal, Vol.15, n. 3, pp. 68-77, 1991.
- [8] L.Tarabella: "*Real-Time Concurrent PascalMusic*", Interface, Vol.22, pp 229-241, 1993.

Capitolo 8

**SISTEMI MULTIMEDIALI,
REALTA' VIRTUALE,
SPAZIALIZZAZIONE**

APPLICAZIONE MAX PER LA SIMULAZIONE DI SORGENTI SONORE IN MOVIMENTO CON DISPOSITIVI COMMERCIALI A BASSO COSTO

Andrea Belladonna, Alvise Vidolin

Conservatorio di Venezia "B. Marcello"

C.S.C. Università degli Studi di Padova
via Gradenigo 6/a - 35129 Padova - Italy
phone: +39 49 8287631 - fax: +39 49 8287699
E-mail: vidolin@maya.dei.unipid.it

Abstract

The system we have realized allows:

- moving sound sources simulation on 2 to 8 independent audio channels systems;
- possibility to have sound movements in every direction;
- capability of the system to be adapted to different kinds of ambients;
- simulation of virtual space dimensions and "near/far away" effect of sound source even out of real space;
- full real-time control capability of simulation models by sophisticated graphic-interactive user interface.

The system is based on a program developed in Max, on Apple Macintosh Computers; it controls a group of MIDI-driven audio potentiometers; automated potentiometers produce the necessary audio volume modification perceived as interaural amplitude variations.

Digital signal processors produce delayed and reverberated signal.

The system can be employed in all those live performances where Live Electronics needs sound source movements simulations, even if very complex.

The structure of program allows a total control of different parameters that defines the virtual space dimensions and sound source movements in it.

The user can control the several parameters in two different ways: in real-time mode and with "playlists" of sound movement events written in a simple internal language.

Introduzione

Il sistema qui descritto è in grado di effettuare la simulazione di sorgenti sonore in movimento su installazioni da 2 a 8 diffusori indipendenti con possibilità di compiere percorsi

su tutte le direzioni. Può essere utilizzato nell'esecuzione in concerto di lavori di Live Electronics, realizzando in tempo reale alcuni modelli di simulazione originariamente sviluppati per il tempo differito [1] oppure implementati su prototipi di difficile reperibilità quali il Sistema 4i [2], l'Halaphon [3] o il Trails [4].

Il sistema si basa su un programma realizzato in ambiente Max, su piattaforma Apple Macintosh, che pilota un banco di potenziometri audio controllabili a distanza via MIDI, i quali danno luogo alle modificazioni percepibili come variazioni interaurali di ampiezza. Processori digitali di segnale generano il segnale ritardato ed il segnale riverberato.

L'utilizzo di controllers MIDI come nucleo di controllo del sistema permette inoltre il controllo remoto e quindi di essere svincolati dalla posizione della consolle di mixaggio; questo consente una notevole comodità operativa e non esclude l'utilizzo del programma da parte degli stessi strumentisti durante un concerto.

Il sistema può trovare impiego in tutte quelle applicazioni esecutive in cui l'aspetto del Live Electronics necessita della messa in opera di spazializzazioni anche complesse.

Rispetto ai sistemi dedicati alla spazializzazione, quali il processore SP-1 [5], l'Halaphon,

il Mini Trails ed altri, questo sistema utilizza elementi discreti di assoluta reperibilità e può adattarsi alle innovazioni tecnologiche che l'industria degli strumenti musicali MIDI offre di continuo; nella versione attuale, presentata nella dimostrazione, è progettato per gestire due Niche Audio Control Module (ACM) e due processori digitali di segnale Yamaha SPX1000.

In alcune prove effettuate i Niche sono stati sostituiti dal sistema Sound Engineer della GeneralMusic [6] con risultati equivalenti.

Il nucleo del sistema, essendo un insieme di patches MAX ognuna delle quali svolge un diverso compito, non risulta quindi essere vincolato ad un particolare hardware ed è in grado di adattarsi ai differenti dispositivi impiegati.

Descrizione del sistema

Il sistema, illustrato schematicamente in figura 1, è composto da:

- 1) un banco di 8 potenziometri controllabili via MIDI (ACM 1) per ottenere gli inviluppi di ampiezza associati a ciascun altoparlante;
- 2) un processore digitale di segnale dedicato alla riverberazione;
- 3) un processore digitale di segnale dedicato alla generazione di delay con tempi di ritardo variabili da 1 a 30 ms;

4) un banco di 8 potenziometri controllabili via MIDI (ACM 2) che consente di controllare il segnale diretto, il segnale

ritardato ed il segnale riverberato eventualmente anche per più linee.

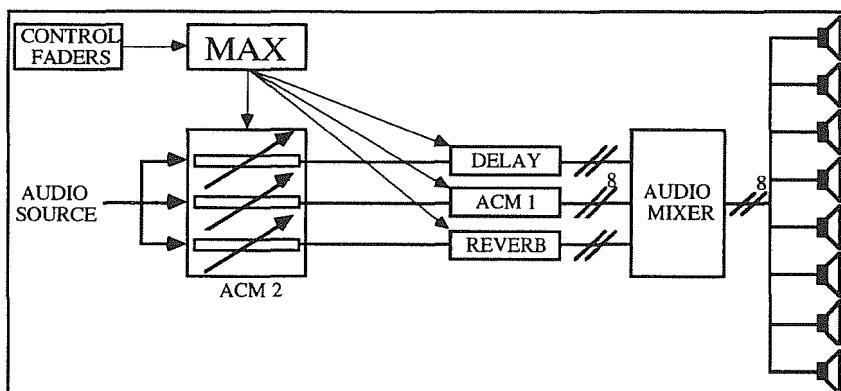


FIGURA 1

Descrizione del generatore di movimento

Il generatore di movimento è costituito da due generatori di inviluppi di ampiezza sincronizzati in modo da ottenere una dissolvenza incrociata tra il livello di ampiezza di un diffusore e quello successivo nel percorso stabilito. La curva di inviluppo è definibile dall'utente in varie modalità (disegno manuale, generazione algoritmica), prevede il controllo di soglia (livello ampiezza minima) ed è visualizzabile graficamente. Questo consente di adattare il generatore di movimento a qualsiasi ambiente esecutivo e di

rendere il movimento più o meno evidente (vedi figura 2).

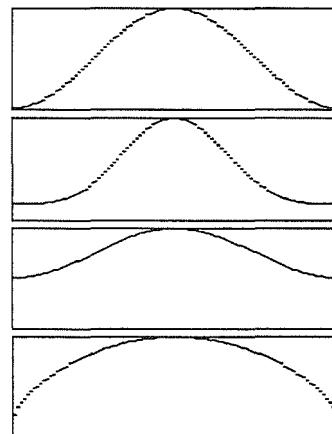


FIGURA 2

La curva di inviluppo è definita in una tabella di 128

punti e viene letta con una frequenza di clock variabile il cui valore massimo è di 200 Hz; questo garantisce una quantità ridotta di dati MIDI da trattare e da trasmettere, unitamente ad una sufficiente risoluzione. Per ottenere inviluppi di durata inferiore ad 1 s mantenendo inalterata la quantità di dati MIDI generata, viene incrementato il passo di lettura della tabella.

Tutte le prove sperimentali effettuate hanno dimostrato accettabili durate di inviluppo fino a 0.1 s, ben al di sotto quindi dell'uso corrente.

I percorsi vengono realizzati associando a ciascun inviluppo un MIDI Control Change differente.

La tecnica della dissolvenza incrociata fa sì che vengano coinvolti solo due controllers alla volta e pertanto è facile realizzare percorsi diversi ed articolati. Il cambio di MIDI Control Change avviene in corrispondenza dell'inizio di lettura della tabella onde evitare disturbi dovuti alla commutazione.

Nelle figure 3 e 4 sono visualizzati due particolari del generatore di movimento.

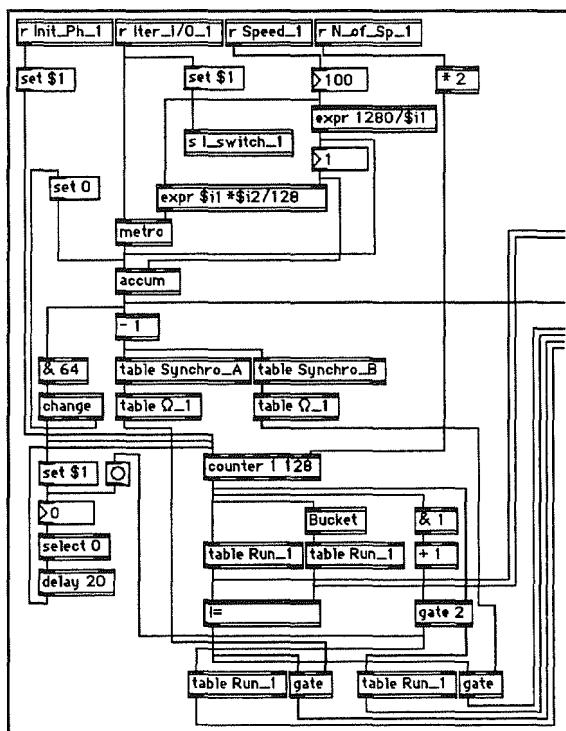


FIGURA 3

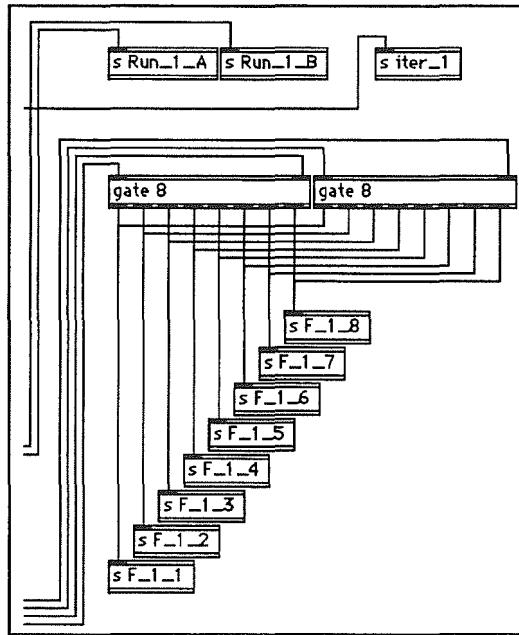


FIGURA 4

Il fatto che siano coinvolti solo due diffusori alla volta provoca un calo di potenza nel sistema generale di amplificazione e una estrema polarizzazione del segnale su una coppia di diffusori. Per compensare questo inconveniente è stata introdotta una linea di ritardo con tempi variabili da 1 a 30 ms, in dipendenza dalle dimensioni della sala, che invia con il livello opportuno a tutti gli altoparlanti il segnale ritardato. Per l'effetto precedenza [7] si mantiene la direzionalità del suono che arriva dalla coppia di altoparlanti attiva, ma l'energia totale del segnale è data dalla somma del segnale proveniente da tutti i diffusori.

Nel caso di rapidi movimenti, le brusche variazioni di ampiezza tra un diffusore e l'altro possono diventare fonte di disturbo; allo scopo di ridurre questo inconveniente fino a livelli accettabili, un apposito algoritmo si incarica di modificare l'inviluppo di ampiezza di ogni altoparlante attenuando la pendenza della curva di aumento/decremento di ampiezza e contemporaneamente innalzando il livello medio di soglia, ossia di permanenza di parte del segnale su tutti i diffusori. In pratica, più la velocità di rotazione è rapida, più ridotta è la variazione di ampiezza tra i vari diffusori durante la transizione del

segnale. Il percorso della sorgente può essere effettuato anche al di fuori dello spazio reale in quanto esiste un algoritmo che determina una opportuna proporzione tra il

segnale diretto e quello riverberato.

In figura 5 è riportato uno schema di spazializzazione a 4 canali.

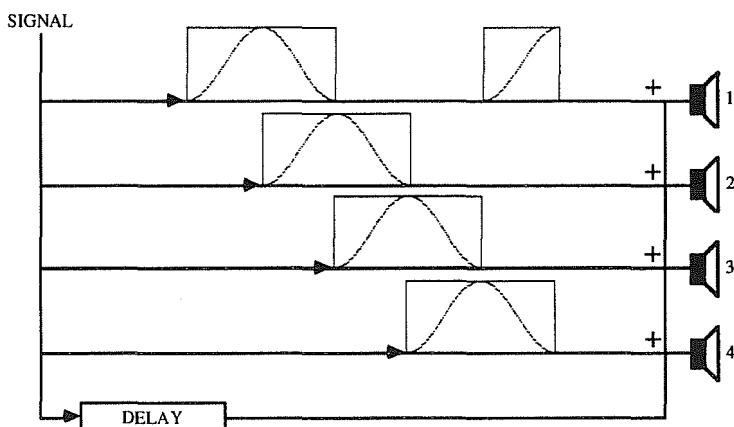


FIGURA 5

Modalità esecutive

Il controllo dei movimenti della sorgente sonora può essere gestito con due modalità operative: "Realtime" e "Playlist".

La modalità Realtime prevede il controllo gestuale della maggior parte dei parametri di gestione dello spazio; la modalità Playlist consente di definire una lista di eventi di spazializzazione che possono succedersi in modo automatico o mediante comando impartito dall'utente.

- Modalità Realtime: in questa modalità il programma prevede il controllo in tempo reale via

dispositivi MIDI di ogni tipo e/o tastiera alfanumerica del computer dei seguenti parametri: attivazione/disattivazione della linea di spazializzazione, livello, velocità, direzione di movimento, allontanamento e/o avvicinamento della sorgente sonora. L'assegnazione di controller MIDI esterni ai diversi parametri del programma è completamente configurabile dall'utente.

- Modalità Playlist: questa modalità prevede di memorizzare su di un file ASCII la successione degli eventi-spazio mediante un linguaggio simbolico. Tale linguaggio è

composto da diversi tipi di istruzioni, fra cui: spostamento del segnale verso uno (o più) diffusori lasciando inalterato il livello degli altoparlanti da cui lo spostamento ha inizio, spostamento del segnale verso uno (o più) diffusori con crossfade rispetto agli altoparlanti da cui lo spostamento ha inizio, rotazione circolare in senso orario o antiorario, percorso aleatorio, percorso complesso iterativo (ossia ripetuto fino a nuovo comando), percorso complesso non iterativo (ossia spostamento della sorgente sonora da un punto ad un altro o allo stesso passando per uno o più diffusori). La sintassi utilizzata dal linguaggio è la seguente:

```
comment, opcode par1  
par2...parN;
```

Il commento è ciò che viene visualizzato durante l'esecuzione e serve per comunicare all'utente ciò che sta facendo il programma; il codice operativo definisce il tipo di evento da compiere ed è seguito dai parametri propri dell'istruzione da eseguire.

Interfaccia utente

Particolare riguardo è stato dato all'interfaccia utente con lo scopo di semplificare al massimo le operazioni di configurazione del sistema e l'utilizzo del programma stesso; tutte le

numerose finestre possono essere richiamate mediante pulsanti e menu a tendina.

Il setup principale del programma nel quale vengono impostati i parametri relativi all'ambiente esecutivo (posizione degli altoparlanti e dimensioni della sala) viene eseguito in modo grafico; durante l'utilizzo del sistema sono disponibili una serie di finestre che consentono di tenere sotto controllo ogni aspetto dell'esecuzione sia con visualizzazione grafica che numerica.

Conclusioni

Nello sviluppo di questo lavoro si è cercato di tenere il programma il più possibile svincolato da apparecchiature di qualsiasi tipo e marca; la struttura modulare del progetto consente inoltre di aggiungere altre patches in grado di compiere nuove funzioni senza dover modificare il software nelle sue linee essenziali. Le varie subpatches dedicate ai diversi compiti sono infatti legate al kernel del programma in modo autonomo. Durante movimenti rapidi della sorgente sonora non è stato preso in considerazione l'effetto doppler in quanto le conseguenti variazioni di frequenza, pur dando un notevole effetto di realismo, diventano fattore di disturbo su un piano puramente musicale.

Riferimenti bibliografici

- [1] J. M. Chowning: "*The Simulation of Moving Sound Source*", Journal of the Audio Engineering Society, 19:2-6, 1971.
- [2] M. Graziani: "*Riverbero e Spazializzazione nel processore 4i*", Quaderno LIMB 4, pp. 41-48, 1984.
- [3] H. P. Haller: "*Live-Elektronik*", Teilton Schriftenreihe der Heinrich Strobel Stiftung des Südwestfunks, pp. 41-46, Bärenreiter-Verlag, Kassel, 1980.
- [4] N. Bernardini, P. Otto: "*TRAILS: An Interactive System for Sound Location*", Proceedings of the International Computer Music Conference 1989 Columbus, Computer Music Association, San Francisco CA, 1989.
- [5] "A Real-Time MIDI Processor for Sound Spazialization", Operating Manual, Spatial Sound Inc., Fairfax, CA, 1990.
- [6] "Sound Engineer", GeneralMusic S.p.A., S. Giovanni in Marignano (FO), 1993.
- [7] B. C. J. Moore: "*An Introduction to the Psychology of Hearing*", Academic Press, New York, 1982.

A SYSTEM FOR REAL-TIME CONTROL OF HUMAN MODELS ON STAGE

A.Camurri F.Giuffrida G.Vercelli and R.Zaccaria

DIST - Univ. di Genova
via Opera Pia 11a, I-16145 Genova, Italy
e-mail: sand@dist.unige.it

Abstract

The employment of human models interacting with actors on stage is often used in science fiction and fantasy movies with sophisticated, three dimensional graphic tools. It is a difficult task to use these techniques in a theatrical environment. In a theatrical performance it is essential to exhibit a *real* interaction between the actor and the model, so that the system is able to react in real-time. Such a system should be multi-medial in a whole sense: it should be able to generate and modify music in real-time influenced by the movements of the actors on stage, and the directives of both the choreograph and the composer. In this paper, a prototype of such a kind of "theatrical machine" is described, for the control of a theatrical (including musical) event.

System Architecture

The animation of a human model is carried out by tracking the movements extracting the relevant points of the human figure, the positions of the joints of the human skeleton, and calculating the kinematic structure to move the model. The real-time tracking of the actor's movements is realized using an advanced acquisition system named CosTel [7] [1] CosTel (Space Coordinates by means of Linear Electrical Transducers), an acquisition and processing system for three dimensional kinematic data designed for use in biomechanics, neurology, robotics and sport medicine. The main feature of CosTel is a high resolution recording capability which allows to simultaneously track the movement of several infrared spot lights, both in space and time.

We can record the spatial trajectory of each independent

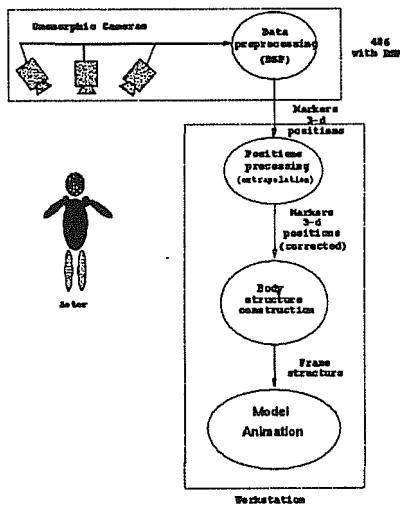


Figure 1: Structure of Acquisition's System

zone by placing the markers in selected zones of the human body. A special vision device formed by three anamorphic cameras generates three streams of monodimensional data. A reconstruction algorithm extracts from these streams the spatial coordinates. The markers consist of infrared light emitting diodes fired sequentially and scanned simultaneously by the three anamorphic cameras, allowing the automatic markers identification. Markers data are transmitted via a special radio link to the host computer, in order to synchronize and merge each data stream coming from the cameras. The total absence of any cable connection between the markers driving unit and the CosTel main unit leaves the actor absolutely free in his movements.

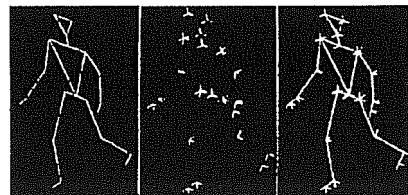


Figure 2: Example of reconstruction

Of course, the actor must play into a restricted area, and as far as possible in front of the cameras, since every camera has to acquire all the markers.

Preprocessing and Extrapolation

In order to have a good real-time tracking of every marker, the system has to *see* each marker from the three cameras: this precondition is often not verified due to the actor's movements during a performance.

Given an actor moving on a stage, it often happens that some markers are not visible from all cameras: so it is necessary to have an algorithm able to extrapolate missing data from the previous measures.

This module (PE) simply keeps in a queue the measures, and when a new incomplete measure arrives (with some markers data missing), it extrapolates the missing data using a standard minimum quadratic distance approximation algorithm..

3-D Reconstruction of the human figure

From the PE module we obtain a stream of frames (normally 15-20 frames per second). Each stream of frames represents the measures calculated by CosTel and processed by the PE module, transformed into a smooth trajectories of the markers. Starting from these trajectories, it is necessary to compute the complete human kinematic structure at each frame computed by the PE module.

So far, the problem is how to move from a rough geometric description (a nebula of 3-d points independently moving, see the central image of figure 2) to a structured kinematic description (constrained to the human body structure, see the image right in figure 2). To reconstruct the human body is necessary:

- link the 3-d points to form the skeleton that represent the man
- compute the spatial-geometrical relations.

To perform these two steps, it is necessary to use a representation of the body: in our system the body structure is based on *kinematic chains of geometrical frames*, where each *frame* is a 4×4 homogenous matrix, describing the position and the orientation of the point. A 3×3 matrix represents the orientation of the target, and a vector represent the position.

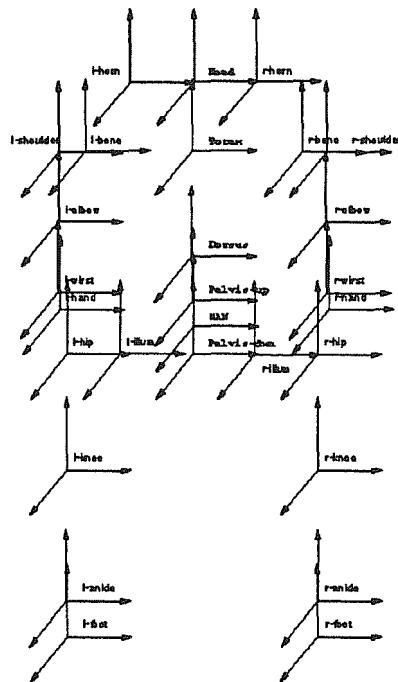


Figure 3: Kinematic human structure

Each matrix represents the roto-translation relation between the considered frame and its relative frame (called the *ancestor*). The *frames* are linked in a chain. Movements, rotations and positions are relative to the ancestors and are not absolute. This is a traditional approach often used for the control of robots in real time [5].

As shown in figure 3 a certain number of frames are attached at every joint of the human body. One frame defines the position of the joint, the others (max 3) represent the degrees of freedom (one for each degree of freedom). In this way we can simplify the

calculus of complex rotations in partial, sequential rotations. The partial rotation are computed and executed always in the same sequence, otherwise the position we obtain is not the one we need. The rotational frames of a joint are all in the same position (the position of the ancestor) and are linked in the sequence in which the rotations are computed and executed. The base frame of the human figure is the *MAN*. The reference frame attached in the pelvis position defines the position of the human body on the stage. *Pelvis-up* and *Pelvis-down* are the first two derived frames of *MAN*, *Pelvis-up* has 3 degree of freedom (x-y-z order), *Pelvis-down* is strictly linked to *l-ilium* and *r-ilium*, the frames linked and used when you need to use only one leg and not all the human structure. *l-ilium* is linked to *l-hip*, this joint has now two degrees of freedom, on the x axis and on the y axis. The complete body is in this way built. It is also possible to use only a part of it, if it is necessary to animate only a leg or an arm. In this case the ancestor of the structure is not yet the *MAN* frame, but the ancestor of the chain used: for example the father of the left leg is *l-ilium*.

After modelling the human body as a tree of kinematic chains starting from the pelvis, it is necessary to fit each geometric frame with the position of the markers onto the actor. Once solved the problem of the positions of the markers on the

actor, we recompute the kinematic transformation. Starting from the root frame *MAN*, and considering the direction of the motion, we calculate the rototranslational matrices going in the two direction of the chain, as shown in figure 4.

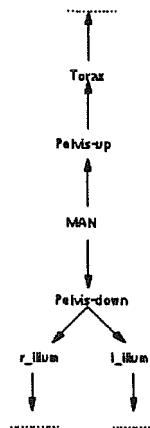


Figure 4: Order of generation of matrices

The final reconstructed human kinematic structure is used to control the graphic model. The graphic model, and also the virtual world is based on the Iris Inventor Toolkit, the software toolkit running on the Silicon Graphics.

The Human Kinematic Algorithm

In this section we refer to the frames defined in figure 3. The Human Kinematic Algorithm builts up the kinematic structure

of the human joints [6] starting from the measures extracted from the markers.

To explain the algorithm we must focus on three base functions:

- Znim - starting from three points A, B, C individuates the plane and calculates the vector V normal to the plane, starting from A
- Xnim - starting from three points A, B, C it calculates the vector (B-A) and normalizes it.
- Ynim - Completes the orthonormal triple build with Xnim and Znim.

The algorithm is the following:

1. Init the frame *MAN* using the marker on the *pelvis*, the *l-ilium* and the *r-ilium*: it calculates with Z min the Z axes orthogonal to the plane individuated by the pelvis, the *r-ilium* and the *l-ilium*, then calculate a second vector parallel to the Z axes of the environment, so it calculates the rototranslational matrix of the orthonormal triple.

2. Init the positions of *Pelvis-up* and *Pelvis-down* as middle point between *l-hip* and *r-hip* at known distance, then calculates the matrices, as translations of *MAN*.

3. Init *dorsum* using the coordinates of the marker on the *thorax*, then calculate the matrix for *dorsum* using those coordinates and the frames just computed.

4. Using the markers on *thorax*, *left horn*, *right horn* and *head*, calculates the position of *head*, and after computes the matrix, keeping the Z axis in the same versus of *MAN*.

The system in a theatrical environment

The system we are developing allows the real-time tracking of a human figure onto a virtual replication. Starting from the current software prototype we plan to use our virtual "puppet" to perform further transformation useful on stage, including the following:

- specular reconstruction
- from the kaos to the man
- modifying the shape of the body part in real-time
- mime on stage, real world on the screen

Specular Reconstruction

Modifying the regular tracking of the movement in to a specular tracking (mirroring the left side with the right side), the actor can compete with his puppet.

From the Kaos to the Man

The animation software is organized in a way that it is possible mount and umount the graphical object on the kinematic structure in real-time. So we can assemble the puppet building the different object starting from the surfaces and nearing the objects

to the place where the puppet is going to be created at a given speed. Since the surfaces of the objects are tracked by the kinematic structure, even if are not on the puppet, it is possible to manage a transformation from a complete kaos situation from which to the image of the moving puppet, is also possible to have {em explosions} of the puppet, or part of it, during the motion.

Modifying the Shape of the Body Parts in Real-Time

This is under implementation, it will be possible to transform any part of the body. Now is possible only to change colors, materials and dimensions of the body parts during the execution of an action.

Mime on Stage, Real World in the Screen

The actor is on a empty stage, but he/she can interact with a puppet that lives in a world full of objects, that he can play, move and pass trough them, so we shall integrate our system with devices as gloves, goggles and other units used in virtual reality applications, to improve the interaction between the actor and the "virtual world" that the puppet lives into.

Music Generation

The data extracted from the CosTel sensing system, and pre-processed by the reconstruction software are embedded in

WinProcne/HARP [2] as analogical modules. Their outputs are also used for controlling computer music synthesis tasks.

From a knowledge representation viewpoint, WinProcne/HARP is grounded on a twofold formalism: analogical and symbolic. The low-level sound representation and data on real-time performance, as well as the CosTel recognition algorithms, are managed by classes, included in an object-oriented concurrent environment (the analogical level of representation). The analogical level is implemented as a set of actors [3]: each actor is hooked to an object in the symbolic level, which is the repository of high level entities, scores, composition rules, high-level descriptions and definitions in general. The symbolic level of representation consists of a declarative symbolic environment, including a multiple-inheritance semantic network formalism derived from KL-ONE [4], extended with temporal primitives and with a typing mechanism.

References

- [1] A.Cappozzo, "Three-dimensional analysis of human walking: Experimental methods and associated artifacts", Human Movement Science, 10:589--602, 1991.

- [2] A.Camurri, C.Canepa, M.Frixione, and R.Zaccaria, "*Harp: A system for intelligent composer's assistance*", IEEE COMPUTER, 24, July 1991.
- [3] G.Agha, "*ACTORS: A Model of Concurrent Computation in Distributed Systems*", The MIT Press, Cambridge, MA, 1986.
- [4] R.J. Brachman and J.G. Schmolze, "*An overview of the kl-one knowledge representation system*", Cognitive Science, pages 9:171--216, 1985.
- [5] G.Vercelli F.Giuffrida, P.Morasso and R.Zaccaria, "*Hybrid architectures for intelligent robotic systems*", Int. IEEE Workshop on Robot and Human Communication, Tokyo, Japan, 1992.
- [6] P.Morasso, M.Solari, and G.Vercelli, "*Software tools for analysis and representation of movements*", Proc. Symposium on Biolocomotion, Formia, Italy, 1989.
- [7] Macellari V. Costel, "*A computer peripheral remote sensing device for 3-dimensional monitoring of human motion*", Medical and Biological Engineering and Computing, 21:311--318, 1983.

THE USE OF 3-D AUDIO IN A SYNTHETIC ENVIRONMENT: AN AURAL RENDERER FOR A DISTRIBUTED VIRTUAL REALITY SYSTEM

Stephen Travis Pope and Lennart E. Fahlén

DSLab - Swedish Institute for Computer Science (SICS)
Isafjordsgatan 26, Kista, S-16428 Sweden
Fax: +46/8/751-7230; E-mail: stp@sics.se, lef@sics.se

Abstract

We have investigated the addition of spatially-localized sound to an existing graphics-oriented synthetic environment (virtual reality system). To build "3-D audio" systems that are robust, listener-independent, real-time, multi-source, and able to give stable sound localization is beyond the current state-of-the-art—even using expensive special-purpose hardware. The "auralizer" or "aural renderer" described here was built as a test-bed for experimenting with the known techniques for generating sound localization cues based on the geometrical models available in a synthetic 3-D world. This paper introduces the psychoacoustical background of sound localization, and then describes the design and usage of the DIVE auralizer. We close by evaluating the system's implementation and performance.

Introduction

Sound plays a very important role in the way humans communicate with each other and how

they perceive their environment. The SICS Distributed Interactive Virtual Environment (DIVE) [1] [2] is a virtual reality (VR) framework for building applications and user interfaces based on the metaphor of presence and interaction in 3-D-space. It uses graphical rendering-visualizers—to display objects in simulated worlds. As an extension of this, we developed a package for the generation of multiple audio signals that have spatial localization cues derived from the geometry of virtual worlds—the so-called "auralizer" [3].

The DIVE is a multi-user virtual reality system that is implemented within a loosely-coupled heterogeneous multiprocessing environment (i.e., a network of different UNIX workstations). Independent DIVE applications (called "AIs" after W. Gibson), run on nodes distributed within a local- or wide-area network, and update a shared (node-wise replicated) object database. This architecture enables the distribution of object animation and interaction processes for load-

sharing between the user input device handling, and graphical rendering tasks.

The auralizer models sounds that take place in the simulated world using a "source/filter/listener" model of sound processing. It can also be used as a stand-alone system for 3-D rendering of complex interactive sound environments. The package allows DIVE AIs to use sound as another output medium, and allows researchers to experiment with the use of 3-D sound environments, allowing, for instance, new kinds of user feedback using different "aural perspectives" on objects as a way to heightening the degree of illusion.

There are several researchers in VR technology who are developing sophisticated models and hardware/software systems for listener-specific or generic real-time 3-D sound spatialization (e.g., [4] or [5]). Also, a number of expensive hardware systems aimed at high-end music studio and film post-production work have been announced recently by major manufacturers. Neither of these domains is the main purpose of our work in the DIVE auralizer, rather we are interested in achieving a "pretty good" 3-D spatialization over headphones or stereo loudspeakers using known techniques and widely available hardware, also adding the requirement that the system run in real-time with multiple (on the order of ten or more) active sound sources. We

want to provide a framework in which one can experiment with models of the perceptual cues involved in sound source localization, and test these in virtual worlds in combination with visual cues and 3D-interaction devices such as magnetic trackers. This document discusses the design of the DIVE auralizer. We first outline the basis for spatialization of sounds in the hearing system and its implications for system design and performance. We focus on the "cheap" psychoacoustical models it uses-derived from the literature of recording engineering and electroacoustic music-for doing model-based sound spatialization. We then discuss the auralizer as a real-time, distributed DSP application. The work described in this report was done as part of the MultiG project, a Swedish national effort at developing very-high-speed wide area networks and distributed software tools and applications.

Synthesis of Localization Cues

There exists a significant literature in psychoacoustics that is devoted to understanding how humans determine the characteristics of spaces using aural cues and how we localize sound sources in a 3-D space. As in several other areas of perception psychology, there are several simple, well-understood mechanisms at play, and other, quite complex and still poorly-under-

stood, ones that are equally important. It is obvious that it is central to our ability to localize sound sources that we have two ears, that they are rather directional in their reception pattern, are separated by a small (and constant) distance, and face in different directions. Deeper reflection leads to the realization that our ears are asymmetrical in both the horizontal and vertical (cutting through the head) planes, and that there are other cues (e.g., Doppler shift, or inter-ear delays), and higher level functions in the brain (see e.g., [6] or [7]) that we use to localize sound sources.

The basic model of sound localization uses direction and distance to characterize the relative geometrical orientation of the sound source and the listener, and provides cues for these, such as relative and absolute amplitude, spectrum envelope and inter-ear delays. Simple reverberation as a cue for the characteristics of the space is also widely used, whereby reverberation time can be keyed to the volume of the space, and the direct-to-reverberated signal ratio is mapped onto the source-listener distance. The more complex relationships between the spectrum of the sound and its location and the characteristics of the space is still a topic of research.

Looking at the basic perceptual features of a sound, we can relate each of them to one or more

aspects of a spatial model as shown in Table 1.

There are several approaches to simulating the cues we use to localize sounds. Some of them are based on direct interpretations and implementations of the physics of sound propagation and the anatomy of our ears—so-called sound ray-tracing techniques that use the "pinna transform," also called the "head-related transfer function" (HRTF)—and others are based on simple psychoacoustical insights. We will present an example of the latter kind of model—the type that we are trying to explore and improve upon in the DIVE auralizer.

- Amplitude (loudness) distance to source
- Position (in 3D space) direction to source
- Spectrum (many dimensions) distance, direction, space
- Reverberation (ratio/characteristic) distance, space, environment
- Inter-aural delay time direction to source (Haas effect)

Table 1: Possible mappings between sound features and spatial cues

Figure 1 below shows the geometrical model and the basic mapping equations used in current psychoacoustical models of localization. The distances—shown as a and b in the figure—between the source and the listener's two ears, which are assumed to be separated by the fixed distance e ,

determine the relative amplitude of the signal that the listener hears. The left-right stereo volume balance is related to the ratio of the difference between $a - b$, and e (shown by the angle Θ in the horizontal $[x/z]$ plane). The amplitude scale factor and direct-to-reverberated signal ratio are proportional to the square of the average distance $d = (a + b) / 2$. The absorption of high frequencies by the air between the source and the listener can be modeled as a low-pass filter whose slope (Q) and cutoff frequency vary proportional to d^2 .

There are several ways to model the spectral changes in a signal according to its direction in the horizontal plane. Making sound sources behind the listener "less bright" than those in front of, or to the side of, the listener is perhaps the most trivial model. This can be simulated simply by a low-pass filter whose Q and cut-off frequency vary (according to some non-linear look-up table) with Θ . The height of a sound source, measured as angle Ψ in the x/y plane, effects the interaural time delay, the sound's spectrum (via the HRTF and possibly the listener's shoulder), and the characteristics of the reverberation. The spectral effects are complex and still poorly-understood, but are simulated in our "cheap" model by two filters: a fairly broad band-reject or "notch" filter in the upper speech range of frequencies (2-4

kHz) combined with a mild boost in the 7 kHz region.

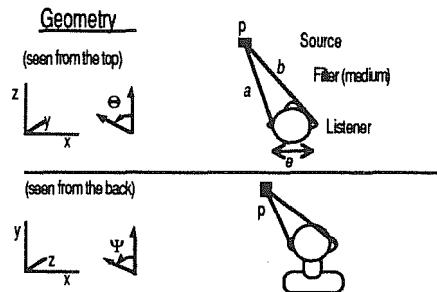


Figure 1a: Auralizer spatial model

The change in the reverberation signal can be approximated by the addition of more early reflections to the signal, simulating echoes off of the walls and ceiling of the simulated environment.

This simple geometrical model has all the necessary information to map geometrical properties onto more sophisticated or higher-level cues, such as more complex filters for the front-back and height cues, and more location-dependent reverberation techniques.

There are many other details of localization, such as what cues we use to differentiate between sounds that have the same interaural time delay-those that lie on the so-called "cones of confusion" that surround each ear and are symmetrical with respect the medial plane (the y/z plane passing through the listener's head between the ears)-and how we determine the difference

Spatial Cues

Distance (d)	= $(b + a) / 2$
Loudness	$\propto d^2$
L/R ratio	$\propto (b - a) / e$
Low-pass filter	$\propto d^2$
Direct/reverb ratio	$\propto d^2$
Spatial low-pass filter ratio	$\propto \Theta$
Spatial band-reject filter	$\propto \Psi$
Inter-aural time delay	$\propto (a - b) / e$
Initial/decay reverb ratio	$\propto \Psi$
(many more possible)	

Figure 1b: Spatial localization cues

between the characteristics of the space and of localized sound sources based on the nature of the signal's reverberation. These are, however, beyond the scope of our present model. The reader is referred to [9] or [10] for more detailed discussions.

Building an Auralizer for DIVE

We will now describe that environment (the SICS MultiG DIVE) in enough detail to provide the background for our introduction of the software architecture of the DIVE auralizer. For a more in-depth discussion, see [1] and [2].

The DIVE Architecture

Most VR systems are mainly concerned with a 3-D rendering of a simulated virtual world, and with user navigation among, and interaction with, objects in this world [8]. In these systems, some "database" exists, along with some data about the "user's" position in the world (the position and perspective from which the world will be rendered). This data is used by the render-

ing component, which generates a visual display. The user interaction, database management, and rendering systems can be relatively independent of each other (as they are in the DIVE), with the interface asynchronously driven by changes in the user's position and the state of the objects in the world each time the renderer generates a new frame of video information for display. The renderer can thus be said to "poll" the world and user state for each frame it displays.

The basic unit of activity in the DIVE is the AI, or application-a UNIX process running on some node in the network that broadcasts change messages via the system's event distribution mechanism. An AI might, for example, represent a clock that updates itself every second, distributing messages throughout the network as to its new visual appearance. Visualization AIs that are in the same world as the clock would receive this message, and update their renderings of it (in case it is in the user's field of vision).

The DIVE architecture allows multiple "virtual worlds" (which may or may not be connected by gateways) to be active on the same network, with one or more users in each of them at any one time. This distributed, multi-world, multi-user, non-server (peer-to-peer) architecture is perhaps the most interesting feature of DIVE. Figure 2 shows

a schematic overview of the architecture. Visualization and interaction nodes manage the graphical output and gestural input for the user. The visualization nodes can potentially support stereo-optic output systems such as head-mounted displays.

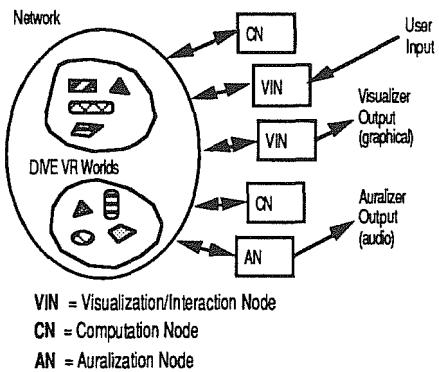


Figure 2: DIVE architecture

Computation nodes can execute AI processes such as ticking clocks, or "supporting" processes such as a collision detector or gravity simulator, which update the state of other objects in the world database without introducing any of their own. The degenerate case is that all of these processes are running on the same workstation, though we use between three and six in common practice.

The Auralizer node shown in the lower-right part of Figure 2 is the component that we have added in this project. Like a visualizer, it will "render" the state of the world object data base, but using stereophonic audio instead of video as its output medium.

The DIVE Auralizer

AI programs "register" sounds with the auralizer by sending out messages that include a unique identifier, a sound file name, and a sound index, and a "perspective" (which can be thought of as a sample name and channel and key numbers in MIDI synthesizer terminology). The AI can later play this sound by sending out a message that includes the id and indices, and the relative amplitude of the sound.

The auralizer runs as a separate process, probably on a network node with high-quality stereo audio output hardware. It maintains a table of the sounds that have been registered by AIs in the current world and responds to sound output messages by playing the chosen sounds, possibly mixing them with other active sounds and spatializing the result to provide for an aural model of the virtual space. To trigger a sound, the auralizer needs to know which sound is being requested, the position of the sound source (possibly in motion), the position and orientation of the listener, and the characteristics of the virtual space. It will use this information to select and process the stored sample. There can be several types or levels of auralizers depending on the amount of computation power that can be dedicated to the auralization (i.e., if it is running on the same workstation as the visual ren-

derer, or if special DSP hardware is used).

The architecture of the auralizer was debated for some time, and several approaches were prototyped. The final design was factored into several components: (1) the interface to the network distribution mechanism; (2) sample structure storage and management; (3) the geometry functions for determining the relative locations of sound sources and mapping the geometry onto the parameters of the mixer, filter and reverberator; (4) the sound mixer, filter and reverberator; and (5) the real-time output handlers for the DACs. The rough architecture of the APE and its interconnection with the rest of DIVE is illustrated in Figure 3. The system runs as three "threads" (light-weight processes): the event, the geometry, and output loops.

When an AI registers a sound, the sample manager stores the sample data of the sound, its duration, rate, format, and other data. The second component is the geometry engine. Each time a client plays a sound, it is necessary to get the coordinates and orientation of the user (defined as the user's "ear" objects), and of the AI in order to determine the relative position and amplitude of the source. This data is updated while the sound is playing by the geometry thread, as both the source and the listener may be in motion.

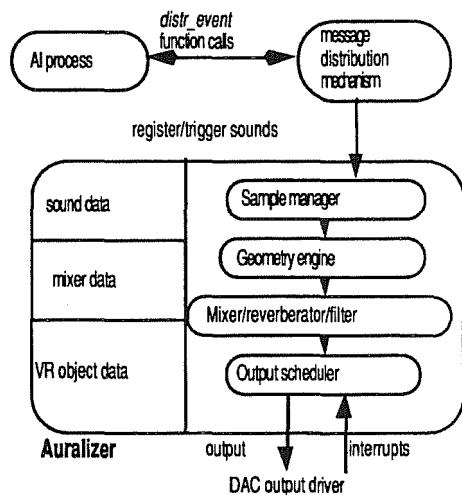


Figure 3: Auralizer Architecture

The sound mixer functions can manage multiple active clients at different locations, and perform the summation of up to 32 monophonic sources into a stereo output stream. The reverberation model adds a small amount of reverberation (mixed according to the geometry engine's data) to the mixed sound array. The DIVE's distribution mechanism communicates between AIs and visualizers. It registers handlers for the collection of callback functions that AIs can call with sound output or other auralizer-related messages. Finally, the real-time drivers and interrupt handlers for the sound output via the DACs are implemented so that different DAC hardware can be substituted with relative ease.

Applications

A number of application areas have been targeted for our testing and further development of the DIVE auralizer. The aim in general-purpose VR-based user interfaces is to increase the naturalness and reduce the cognitive load of the interface. The initial applications were various ticking clocks and bouncing balls. These were used to debug both the system and its initial spatial models. More complex applications to date include an integrated system using the ANIMA choreography [11] system together with DIVE visualizers and auralizers for dance.

The system will also be used in teleconferencing systems to give cues about activities that goes on "off camera" such as participants entering and leaving the conference. We hope to be able to build more fully featured environments for artistic-aesthetic expression, i.e., sound sculptures, music compositions, virtual instruments and interactive experiences such as films and games. A related project aims to make it possible for sight-impaired persons to participate in activities in synthetic environments.

Evaluation

The DIVE auralizer was developed through several phases during the Summer of 1992. The initial platform was a Sun SPARC-station-2 with Ariel Corp. digital-to-analog converters. It was tested in a DIVE

network running multiple sound-generating AIs on various nodes. We implemented the spatial cues of loudness, inter-aural balance, directional filtering in the horizontal plane, and simple reverberation (fixed reverberator parameters with mixing of direct and reverberated signals). The system was found to support sustained real-time output with between five and ten AIs making noises and to provide a "reasonably good" spatial model, with the expected problems related to the "cones of confusion" and missing height cues. The system was ported to the Silicon Graphics Indigo and Sun SPARCstation-10 platforms in April of 1993, where it ran significantly faster and needed no additional hardware for sound output (on the Indigo).

The first auralizer that was built mixed 16-bit 8 kHz monophonic sound samples into a stereo aural image. The current generation runs at 16 kHz and incorporates better stereo reverberation and inter-aural time delay. More sophisticated filters (modeling the HRTF more closely), and dynamic reverberation algorithms are planned.

We have also avoided the issue of AIs that generate their sounds in real time by requiring them to pre-register sound samples. We do this because we do not want to have to address the issues of network bandwidth with multiple real-time sound sample streams, or of abstract timbre description

languages, in the present system, which is intended for experimenting with localization models. The experimental ANIMA system uses disk-based samples that may be too large to fit in memory, but still cannot handle real-time sample streams passed over the network.

Due to delays in the network distribution system, and the relatively slow (10 Hz) frame rate of visualizers, exact synchronization of visual and aural events is difficult. We are currently investigating ways to improve this.

It remains to be seen just how good we can make our spatial models and maintain real-time performance on "reasonable" hardware. The auralizer is written entirely in C and does not use the DSP coprocessors of the platforms it runs on. More complex reverberators and spatial filters can be expected to "push the envelope" of this restriction.

Future Work and Conclusion

As mentioned above, because the auralizer is being built in the context of a project whose goals are understanding distributed programming on very-high-speed networks, and the development of new techniques and tools for supporting this, it was not our intention to place a great deal of emphasis on the exact quality of the spatial model. The auralizer was designed for maximal "pluggability" so that

more sophisticated modules can be substituted after this initial implementation. Another design criterium was to have loose coupling between the auralizer's components, so that they can themselves be distributed over a network in various fashions. Examples of the "pluggability" are the ports of the output stage mentioned above, moving the sample storage to an object-oriented database management system, and the running the mixing, reverberation and spatial filtering components on a special distributed real-time signal processing framework.

We hope to build better geometry subsystems, fancier spatial models, and distributed mixing components in the future while retaining the basic auralizer design. We also think it is important to build, as soon as possible, non trivial applications and have users experiment with them in order for us to gain experience both from the "systems" standpoint and from a "user interface" standpoint.

References

- [1] L. E. Fahlén: "*The MultiG Telepresence System.*", in Proceedings of the Third MultiG Workshop. Stockholm: Royal Institute of Technology, 1991.
- [2] C. Carlsson, O. Hagsand: "*The Architecture of the MultiG Distributed Interactive Virtual Environment.*", Proceedings of the Fifth MultiG Workshop. Stockholm: Royal Institute of Technology, 1992.
- [3] S. T. Pope, L. E. Fahlén: "*Building Sound into a Virtual Environment.*", in Proceedings of the Fifth MultiG Workshop, Stockholm: Royal Institute of Technology, 1992.
- [4] D. R. Begault: "*Challenges to the Successful Implementation of 3-D Sound.*", Journal of Audio Engineering Society 39(2): 864-870, 1991.
- [5] E. M. Wenzel: "*Localization in Virtual Acoustic Displays.*", Presence: Telepresence and Virtual Environments 1(1): 80-107, 1992.
- [6] J. Blauert: "*Spatial Hearing*", MIT Press, 1983.
- [7] Durlach, et al.: "*On the Externalization of Auditory Images.*", Presence: Telepresence and Virtual Environments 1(2): 251-257, 1992.
- [8] S. Helsel, J. Roth eds.: "*Virtual Reality Theory, Practice*", and Promise. Addison-Wesley, 1991.
- [9] J. Chowning: "*The Simulation of Moving Sound Sources.*", Journal of Audio Engineering Society 19: 2-6, 1971.
- [10] F. R. Moore: "*Spatialization of Sounds over Loudspeakers.*", in M. V. Mathews and J. R. Pierce, eds. Current Directions in Computer Music Research, MIT Press: 65-87, 1989.
- [11] T. Ungvary, S. Waters, P. Rajka: "*Nuntius: A Computer System for the Interactive Composition and Analysis of Music and Dance.*", Leonardo, 25(1): 59-68, 1992.

Capitolo 9

STUDIO REPORT

CENTER FOR ART AND MEDIATECHNOLOGY KARLSRUHE THE INSTITUTE FOR MUSIC AND ACOUSTIC

P. Dutilleux

Ritterstr. 42

D-76137 Karlsruhe, Germany

Tel: +49 (0) 721-9340 300

Fax: +49 (0) 721-9340 39

E-mail: music@zkm.de

The Center for Art and Media Technology in Karlsruhe is dedicated to art and its relationship to new media. The center is comprised of the *Museum for Contemporary Art*, the *Media Museum*, the *Institute for Image Media* and the *Institute for Music and Acoustics*. Though the center is in the process of building up its art collection it already offers studio space. The center will be fully operational by the end of 1997, when the new building is completed.

Aside from the collection and presentation of art works, the center places its main emphasis on the production of artistic works in the various institutes. Though the main tool for artistic production is the computer, digital technology does not necessarily have to be part of all works.

The Institute for Music and Acoustics

The Institute for Music and Acoustics is open for artistic and scientific work in general areas of:

The production of musical works (tape, live-electronics, sound installations with or without other media) ;

Open processes for learning and discussion (reflection of esthetic conditions) ;

The development of computer music environments and systems ;

Scientific research (psycho-acoustics, basic musical research).

Multimedia projects are also possible through cooperation with the Institute for Image Media. The two institutes are located in different buildings until the opening of the new center. The Institute for Image Media is situated in a former storage building and offers shop space and – within limits – a performance space.

Under the direction of Jeffrey Shaw, the Image Institute focuses its efforts on computer animation, the construction of interactive installations and architectural simulations. It also offers a video postproduction studio.

Facilities and Equipment of the Institute for Music

Until the new building is opened, the Institute for Music and Acoustics occupies two stories of an apartment building. Space is available for work studios (no recording studio), offices, seminars, a small electronics shop and a MIDI studio. Room for scientific research is limited. A disk-based sound editing system (Studer Dyaxis) is available for general editing and postproduction purposes, including radio play productions, final mixes of material generated with MIDI equipment or NeXT computers, and for audio postproduction of video. Individual CDs can be produced with the Sonic Solution CD Printer.

When the new building opens, the studios of the music institute will include a big control room, one large recording studio (250 square meters), four smaller studios for musical work and scientific set-ups, five rehearsal studios for composers and instrumentalists as well as offices for guests.

Activities and Projects

The Institute for Music and Acoustics contributes to ongoing activities of the center, such as competitions and awards, symposia and exhibitions, concerts, performances and the MultiMediale, a biannual festival. The music group also offers regular workshops for composers and

musicians.

The Open Music System

The Institute for Music and Acoustics is emphasizing the development of an open computer music environment for composition and sound synthesis. This environment is centered around Common Lisp Music by Bill Schottstaedt for sound synthesis and processing and Common Music by Rick Taube for score generation and algorithmic composition. This environment is being developed as a joint venture between ZKM and the Computer Music Center at Stanford (CCRMA). The software is optimized to run on the NeXT computers used at Stanford and ZKM, but is basically machine independent.

Live-Electronics

With the ISPW (IRCAM Musical Work Station) and the NeXT Computer the work for live-electronics is as well emphasized. Composers and performers get support for developing computer instruments that can be played either during the composition process or on stage. We aim at enhancing the ability of the computer at having a musical behaviour (score following, user friendly filters).

Bali Project

In collaboration with musicologists, we develop a system for the analysis of the balinese, music (acquisition of gender playing).

IDEAMA

An important project at the Institute for Music and Acoustics and the Mediathek of the ZKM is the establishment of an archive for electro-acoustic music (IDEAMA). This international archive will store digitally electronic music from its very beginnings along with documenting materials. IDEAMA is set up in cooperation with CCRMA, Stanford University, and with the help of internationally reknowned experts.

Visitation

Our studio space is currently able to support three to four guest artists working in parallel. This number will increase drastically in the new building. Working visits are usually between two weeks and three months; longer periods for learning and working can be supported with stipends. The center offers two Siemens stipends each year, one for music and one for image media. In addition, we are able to finance individual projects with grants. Third party funds raised by the artists are also welcome - especially if a proposed project needs extensive financial support for special instruments or performances. Students in natural sciences can get involved in art-related technical projects during training periods (six months typical).

The Work Environment

The working procedure in our studios is not predetermined. We expect visiting musicians and composers to be willing to work with the machines directly. There are no technicians who will translate or implement artistic descriptions. Our staff will assist with technical introductions and supply assistance when problems arise. Collaborative working arrangements are possible: e.g. a composer and a software specialist may apply as a team to realize a project at the center, or people with different qualifications may join together to work on a project.

The Staff of the Institute for Music and Acoustics

The Institute currently employs nine people, six full time and three part time. The director of the Institute is Johannes Goebel. The staff consists of Pierre Dutilleux (signal processing and acoustics), Rick Taube (software development and workshops), Heike Staff (project selection and concerts), Caroline Mössner (secretariat), Frank Schweizer and Gerhard Wolfstieg (introduction to the studios and project accompaniment), Sukandar Kartadinata (hardware maintenance); responsible for the IDEAMA is Thomas Gerwin (Mediathek).

SOME OBSERVATIONS ABOUT S.V.E.M. ACTIVITIES AT THE CENTRAL ENGINEERING LIBRARY OF GENOA UNIVERSITY.

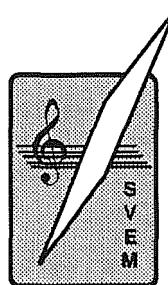
Leopoldo Gamberini - Stefano Mosca
c/o Biblioteca Centrale Facoltà di Ingegneria Univ. Genova
Via Montallegro 1
16145 Genova (ITALY)
Tel. +39 10 3532545 - Fax +39 10 318709

From a bibliotecnic and publishing point of view, a computerized system can give to any musical work better graphics than handwriting and also a lot of opportunities for analysis and musicology.

Through a collaboration between Prof. L. Gamberini (teacher of History of Music at Literature and Philosophy Faculty), the Central

Engineering Library of Genoa University and S. Mosca a search program related to the use of personal computer has been developed; Its aim is musical creation and writing.

The Central Library, already active in using computers for its activities, has arranged, for teachers and students, S.V.E.M. which is the System for Music desktop publishing.



Biblioteca Centrale Facoltà di Ingegneria

S.V.E.M.

*Servizio
Videoscrittura
Editoria
Musicale*

Via Montallegro 5/4 16145 Genova - Tel. 010/308616

System is structured as below:

- 1 APPLE MACINTOSH II ci
- 1 PORTRAIT DISPLAY (A4)
- 1 APPLE LASER WRITER IIIf
- 1 APPLE SCANNER
- 1 APPLE CD Rom
- 1 DYNAFILE DRIVE for 5,25", 1,2M
- 1 LETTORE Syquest 44 MB

- 1 PROTEUS/1 E-mu System
- 1 PROTEUS/2 E-mu System
- 1 Digital Sound Processor YAMAHA SPX50D
- 1 MIDI YAMAHA CLAVINOVA CLP 260
- 2 APPLE MIDI INTERFACE
- 1 Amplification and Mixing system

Professor Gamberini , mostly concerned with the musical aspects of the matter, offered his work "Cristoforo Colombo 12/10/1492" and his experience of composer,musicologist, and conductor .

The work is a Stage Cantata for Barithone Chorus and Orchestra proposed by Genoa University in occasion of the Columbus Celebration for America Discovery, and already performed with good success during Symphonic Season of the City Theater in Genoa.

the text is taken from "Giornale di bordo" by C. Columbus and was realized by L. Gamberini and C. Cormagi.

Considering of S.V.E.M. performances and potential users, the first problem to solve was to realize a specific configuration for manipulating, without difficulties, a large score as the examined one.

The necessary requirements are:
1) Video layout of the complete orchestral score (26 staves in this case) with a more possible complete codification of the western music writing system.

2) A good user interface to write and read score in a simple way also for not "computer competent" users.

3) Possibilities for several modifications on the edited score to satisfy all needs of work personalization. This fact is not negligible considering great evolutions of musical notation

from the past to contemporary experience.

4) Good final product using laser printers with high resolution.

5) Capability of automatic extraction of single parts from the complete orchestral score.

There is also capability of execution of the score realized by musical electronic instruments MIDI connected to computer.

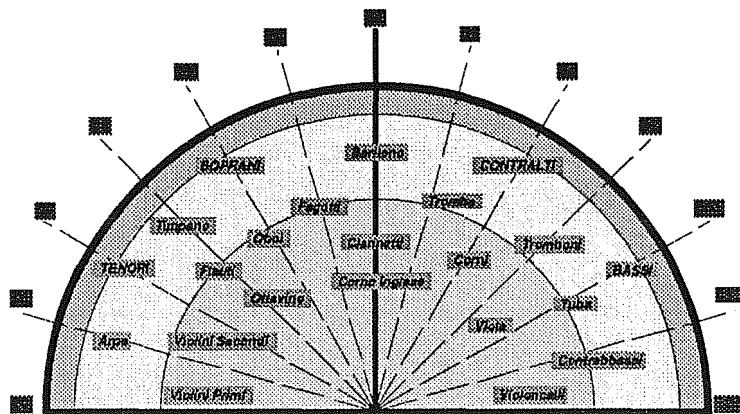
Sound generation is realized by politimbric modules using traditional synthesis methods or multisamples of real instruments.

In this type of realization is necessary to put in evidence some important positions:

1) We don't want to attach any "absolute artistic value" to electronic execution of the piece. This realization doesn't want to substitute traditional performers as conductors any way. Computer can't emulate (at this level) a traditional execution without undermining final result even if there are a lot of margins for correction. Computer has no human feel so emotional presence during execution is missing.

2) Electronic performance can't realize 100% of the score. "Cristoforo Colombo 12/10/1492" was not written as electronic composition so Barithone and Chorus parts are not executed as singing text but as melodic vowels by digital instruments.

3) Electronic execution has principally teaching porpouse for music students; they can listen and modify quickly orchestral exercises.



All the orchestral score has been rewritten with "Finale 2.6" (Coda Music Software). From this edition are pulled out a graphic edition and a executable version in Standard Midi File form. This S.M.F. has been subsequently improved from musical point of view with "Vision 1.4" (Opcode Systems Inc.) obtaining a large document (about 500 Kb on 26 midi channels for a duration of 42 minutes).

These types of experimentations have been already illustrated in some demonstrations.

During Concert Lessons of History of music organized by Literature and Philosophy Faculty of Genoa in May 1992, they demonstrated the capabilities offered by computerized Svem System performing pieces taken from the same opera.

On the occasion of inauguration of the Academic Year 1992 it realized the concert conference at

"Aula Magna" of Genoa University.

Mass media were very interested in use of Personal Computer in Music and promoted a constructive debate between technicians and musicians.

Afterwards were realized some performances of the complete opera (as Symphonic Poem) on the occasion of the recent Columbus Celebration at Scientific Community Stand inside International EXPO of Genoa in July 1992.

For the composer computer becomes a good instrument in writing and publishing and simulator of sound models.

These possibilities that few years ago were own only of Phonology studios and Informatics Laboratories are now available at low cost on account of improvement of personal computer capabilities.

So a large number of musicians, professional and amateurs, were

interested in electronic equipment in reason of simplification of software graphic interfaces.

Also traditional musician, without informatic abilities can simply use computer.

Is perhaps in this circle that precise technical competences of Genoa Engineering Faculty (already working towards multimedia images and digital sound) are complementary to artistic interests realizing an interdisciplinary seldom verified between different branches of learning like music and informatics.

Structure "LIBRARY" as well as place of conservation and consultation for books and magazines tends to become a bibliotechnic services in a broad sense come up again creative centre of a cultural technical artistic world like from time immemorial.

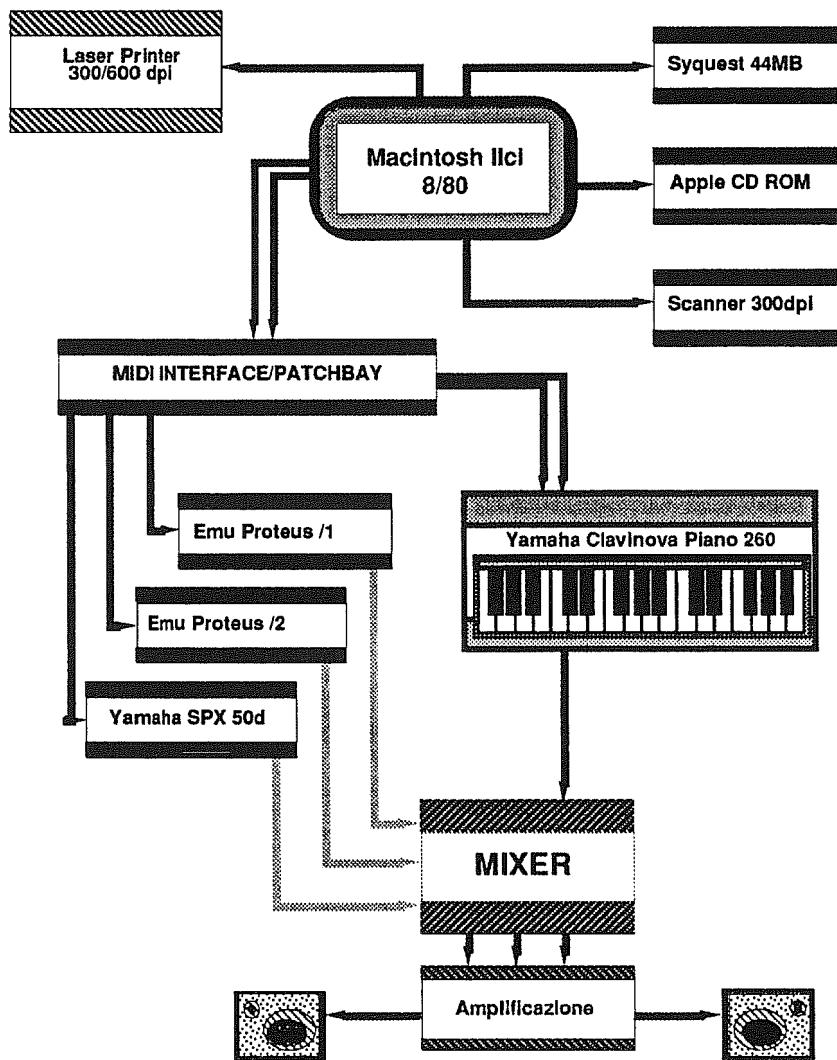
The activation of new informatic services like book list management, bibliographic Database (on CD ROM and on-line consultation is an example of thus tendence.

Another example is the recent inauguration of a new "Informatic Classroom" with a Macintosh network in free use for University students.

SVEM project, on the other way is evolving in a multimedia prospect; infact we can have graphic realization of same original manuscripts kept at Conservatorio N. Paganini in Genoa and Hypertexts

development in collaboration with CNR.

Configurazione Sistema S.V.E.M.



L'ATTIVITA' DELL'ASSOCIAZIONE RICERCARE ED IL SUO STUDIO PER LA RICERCA MUSICALE E ARTISTICA

Lucio Garau, Giorgio Tedde

Associazione RICERCARE
Via Ninasuni, 48
I-09048 Sinnai (CA)
tel-fax 070/765967

Abstract

"Ricercare" is a cultural association born to offer a technical and cultural basis for the production, circulation, and enjoyment of contemporary art and music in Sardinia. The initial fundamental goal of association was that of maintaining the values of the traditional culture in the contemporary artistic language. To fulfil this goal we proceeded along several direction: workshops, concert, commissions of musical research and compositions.

After its first year of activity, the association has now focused its goals. To support this activity, a laboratory has also been created for musical and artistic research. The lab is equipped with state-of-art machinery for digital sound and image processing , and for the editing and pre-printing of musical scores. The laboratory is currently open only to members of the association; however, it will soon be moved to a new centre where it will be possible to host visiting researchers.

This is the last year activity of the association.

- * Research stages and workshops.
- * Research commissions for:
 - * analysis of contemporary music, such as mainstream, popular, and traditional Sardinian music;
 - * analysis of problems related to musical listening;
 - * composition;
 - * recordings of musical and acoustic material.
- * Collaborations with other European musical research association.
- * Shows and concerts to support and promote its work.

L'Associazione Ricercare è sorta per offrire con la propria attività di ricerca un supporto teorico e culturale alla produzione e alla diffusione e fruizione della musica contemporanea in Sardegna. Dopo un anno di attività l'associazione ha potuto focalizzare i propri obiettivi e quindi creare uno studio per la

ricerca musicale ed artistica in generale.

Partita con il proposito di recuperare i valori della cultura tradizionale per mezzo del linguaggio artistico contemporaneo, ha sviluppato la sua attività sia attraverso laboratori-seminari, concerti, commissioni di ricerca e di composizione, che con l'allestimento digitale di suoni e immagini e per l'edizione e la stampa delle partiture.

Diverse esperienze portate avanti sono direttamente collegate allo sviluppo dell'informatica musicale, come il laboratorio-seminario di *Editing di partiture con il computer*, o la produzione di Computer Music per concerto e per teatro-danza.

L'attività dello studio è per ora riservata ai soci, ma è imminente il trasferimento in locali idonei ad accogliere collaborazioni residenziali con studiosi e ricercatori esterni.

L'attività di questa associazione si è incentrata nell'organizzazione di:

- * Stage formativi e seminari di studio-ricerca tesi a coinvolgere i partecipanti in diversi momenti ora rivolti alla semplice informazione-acquisizione di nuove tecniche e informazioni, ora rivolti alla elaborazione-riflessione di queste.
- * Commissioni di ricerca di:
 - * analisi di opere di musica contemporanea colta,

popolare e della tradizione etnica della Sardegna;

- * studi sulle problematiche della fruizione con particolare riferimento alla diffusione della musica contemporanea presso il pubblico;
- * composizione di opere musicali connesse ai lavori di ricerca di cui sopra
- * registrazioni di vari materiali acustici e musicali e studio delle relative tecniche;
- * organizzazione di seminari di studi al fine di diffondere e pubblicare i risultati delle ricerche commissionate ed integrarle con interventi di studiosi e ricercatori in collaborazione con altri enti culturali.
- * Inserimento dell'associazione in un programma inter-associativo con alcuni importanti centri europei di ricerca operanti nel settore.
- * Realizzazione di manifestazioni di promozione e divulgazione delle attività prodotta per mezzo di seminari concerto e installazioni multimediali.

Nel 1991 è stato organizzato un seminario sulle tecniche di scrittura ed edizione musicale con il computer. Sono state presentate le nozioni di base del sistema Macintosh e successivamente le caratteristiche del programma finale. In collaborazione con il docente è stato avviato un

programma di lavoro teso a formare delle competenze di scrittura musicale col computer in Sardegna, ed a migliorare le condizioni di avvicinamento dell'utente a questo programma. Alcuni partecipanti hanno potuto vedere realizzate a stampa alcune pagine delle loro partiture.

Sono state affidate sei commissioni di ricerca ad altrettanti musicologi e compositori e tre commissioni di composizione e tre commissioni di composizione di musica a tre musicisti di chiara fama: Mario Bertoncini, Enrico Correggia, Thomas Kessler.

Questo lavoro di ricerca si concluderà con la pubblicazione e la diffusione dei saggi prodotti, e con l'esecuzione delle musiche scritte inserite in un programma inter associativo.

L'associazione RICERCARE ha già vissuto una fitta rete di relazioni e di scambi con diverse istituzioni analoghe e centri di ricerca europei.

PARTECIPAZIONI

Lyon convegno europeo di acusmatica 1991
Basel Musikakademie 1992
Cagliari Spaziomusica 1992
Cagliari Ente lirico 1992
Basel Musikakademie 1993
Tokio ICMC 1993
Boston Alea 3 1993
Cagliari Spaziomusica 1993

COLLABORAZIONI

Elektronische Studio-
Musikakademie Basel

LAVORI PRODOTTI

M. Bertoncini: *Chanson pour instrument a vent.*
L. Garau : *Canoni.*
Preludio.
G. Tedde: *ATM.*
Cellovoce.
Vox.

ATTREZZATURE

Quadra 700 20/80
Monitor Apple 21"
Centris 650 32/120
Monitor Apple 17"
Digital Film
Hard Disk Blizzard 1 Gb
StreamerDat PLI
PowerbookDUO2308/120
Digidesign Pro-tools
Sample Cell 8 Mb
Piatto Technics
Doppia Piastra Teac
2 Microfoni Schure sm 81
Mic. Sanken COS11BP
DAT Sony
DAT Teac DAP20
Tastera Fathar
MIDI Opcode studio 4
CD-rom Apple
Mixer 16 canali Alesis
Mixer 12 canali Soundcraft
Spirit Folio
4 casse Yamaha NS 10M
Abaton Interfax 24/96
Macintosh Classic
Stampante HP Laserjet 4M
Stampante Style Writer

SOFTWARE

Finale 2.6.3
Alchemy 2.23
Max 2.5
Studio Vision 1.4
Turbosynth 2.0

Capitolo 10

COMPOSIZIONI MUSICALI

THE EFFECT OF DIGITAL SYNTHESIS LANGUAGE ON THE CONCEPTION AND PROCESS OF COMPOSITION

Ludger Brümmer

Visiting Composer at CCRMA Stanford University USA

New Address: Hohenzollernstr. 66

45128 Essen / Germany

E-mail: ffh004@vm.hrz.uni-essen.de

1. Creating tape music with electronic synthesis
The most obvious difference between composing for traditional instruments and digital technology lies in the limitations brought about by the process of performance. The computer has limitations too, but on the whole it seems to be more powerful than the instrumental player as it does exactly what the user tells it to do. And this is the problem! Presented with a couple of notes to play, an instrumentalist would start to add the usual interpretational habits of vibrato, phrasing, dynamic changes, note timbre variations and so on. This adds a complex set of parameters to the composed material and with this a language of convention in the process of musical communication. Their absence in computer music is one of the main problems that the listener encounters.

The appearance of the players on stage, the use of well known instruments and a common expectation of their sound quality (or range of sound qualities), as well as the use of a compositional language help us to perceive and evaluate musical structure.

Music as a kind of communicative language needs well defined expressions if even the surface of its structure is to be perceived. Considering this lack of performer interpretation, the problem for the composer of computer music, is therefore to create a composition with enough new and traditional information. One way of dealing with this problem is to define or redefine the musical material by applying conventions, like instrumental timbres, classical form schemes, equal tempered scales, melodies, rhythms and so on. Another way is to use a new abstract element and expose it in the piece like a motif.

2. The theme of the piece "La Cloche sans Vallees."

One element exposed in the piece "La cloche sans Vallees" is the source piece "La valle des cloches" for piano from the cycle "Miroirs" composed by Maurice Ravel. The whole piece is generated from this sound. Another idea can be described as an "identical or modified repetition of a section of a sound": a loop. Changing the duration of a single loop (pulse) leads to the following types of per-

ception: i.) The shortest loop is perceived as a pulse, or a click without any timbre or therefore pitch. ii.) A longer loop enables the auditory system to, in effect, apply a Fourier Transform to the signal and evaluate at least a part of the timbre of the source signal. iii.) If the loop is longer still, the ear can perceive timbre and pitch completely.

These phenomena also apply to continuous sounds containing repetitive portions of signal and silents.

Human perception structures sound information in the sub audio domain into time units of different size e.g. events, sequences, and sections, while sound informations in the audio domain is perceived as pitch and timbre. Once placed in one of these categories the sound information is perceived. A process of decreasing or increasing speed causes a modification in the perception of a signal as the category crosses the threshold between sub audio and audio perception and with this the whole set of categories moves.

Nearly all the sounds used in this piece create a ritardando through different algorithmic procedures. Even the form itself is an inverted ritardando (or accelerando) followed by another ritardando. The whole structure of the first 10 minutes is transposed or accelerated successively 7 times until it shrinks into a click which again starts to expand, discovering its new contents. At the point where time is compressed to its

limit (the click) the melodic structure of the piano sound is revealed and mirrored, making reference to the symmetrical structure of the source piece.

This technique of discovering material by decelerating the speed is used to connect Ravel's piece with the new composition.

3. The looping program.

The looping program which is created for the above mentioned ideas executes basically pointer operations. Four main functions are specified by envelopes: 1. loop length, 2. reading position in the source file, 3. silent time in between the loops and 4. glissando.

To have access to smaller events as well, some functions determine the individual loop. Each loop can be read forwards or backwards as specified in a list e.g.: (r f f f r r f). The pattern of the list is then repeated.

Each loop is separately transposable with a pattern given in the loop pitch list: (1 1.2 1.32 1 ...).

The idea of the parameter determination is to specify parameters on different time levels, so that a heterogenous global structure can be generated. The pitch parameter, for example is determined 1. globaly with a pitch constant, 2. flexible with the pitch envelope and an envelope scaler, 3. specific in the loop pitch list and 4. by the envelope which applies silents between the loops.

A zero crossing detector with settable limit values and two

ramping mechanisms which fade the sound in at the beginning and out at the end of each loop helps to avoid click. An envelope controls the intensity of this process. The loop and the deceleration of note patterns represent the central idea of the piece. A software synthesis language can now provide a totally different conception of composition. The instrument is not only the source of the sound and carrier of the musical structure (like a conventional instrument), it can represent as a sound synthesis program the musical concept of the piece every bit as much as the musical material itself. The creation of the synthesis program as a creative part of the composition process influences to all the following work and determines the composition. In instrumental music the composition is determined by the technical abilities of an instrument and of its player. But while the abilities of player and instrument could be seen as inflexible constants, the program created for the sound synthesis represent a variable which could be an interactive part in the process of composing.

4. Conclusion.

The tradition of instrumental music provides a notation language. The notation of a new sound is derived from the traditional context of an existing set of sound characteristics and playing techniques. Referring to the process of creation makes it

easier to categorize the sound. The amount of useful sounds which the instrumentalist is able to create with his or her instrument is huge, but all of these sounds are shaped by the instrument itself (material, resonance body), the technique of sound creation (bowed, plucked, blown etc.), the acoustic of the room and the distance of the player from the listener. The limitation of the sounds by the way and technique of production, and perhaps even the optical process of creating the sound, enables the listener to build categories of perception. These categories help us to use a specific sound as an expressive element of a composition, not so much as an effect, but more as an integral part of the composition that can then be varied to a greater or lesser degree.

The way a sound is created in a more or less complicated synthesis process inside the computer makes it difficult to control and categorize the results. There exists no optical representation of the process of production which could help in the perception of the music. Sounds cannot be categorized by the technique used to create them. The same technique applied to different sources can result in widely varying sounds so that it may not be possible for the listener to perceive of these technically linked sounds as linked in an aural sense. Hence the process of structuring is the responsibility of the composer. He or

she has to select the techniques and results and apply limits to the material as a whole, so that some elements are restricted while others are exposed to create the potential of expression for the otherwise unordered timbres.

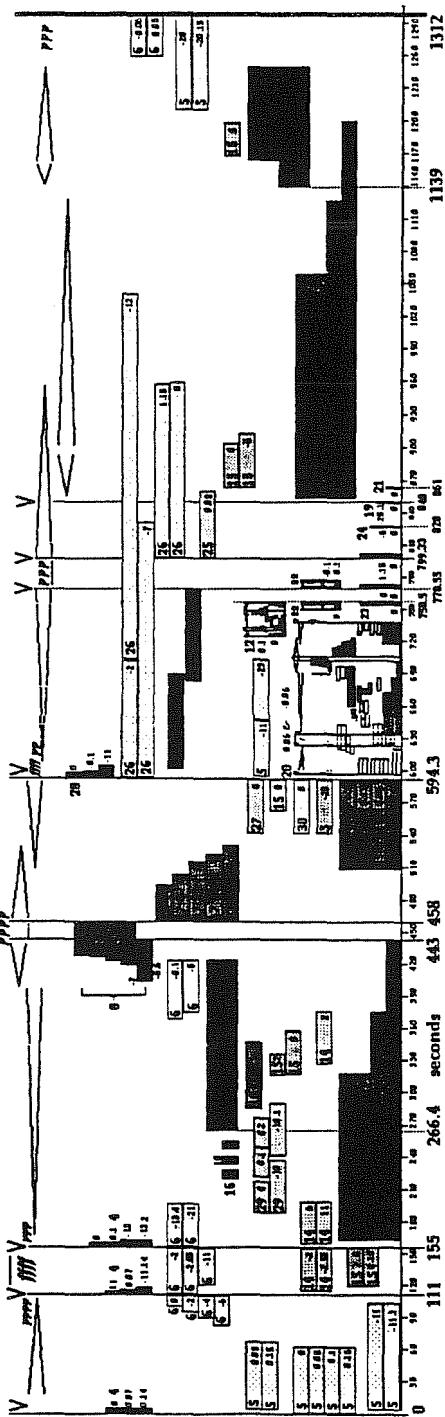
The lack of tradition in computer music, and the explosion of techniques made possible by the computer requires of the composer an additional control and clarity of intention in order to create an expressive musical language. It also requires a knowledge of technical, psychoacoustic, and programming skills to create a piece of music and not merely a piece of sound.

No established and elaborated technique of composition exists as yet in the field of computer music. The topic of "musical language" is the weak point of this genre, especially of music created by newly defined processes of synthesis. The freedom and infinite possibilities of computer music is at the same time both its weakness and its attraction.

The piece was created on the NeXT Computers at CCRMA, Stanford University USA, using William Schottstaedt's Common-Lisp-Music synthesis language and Richard Taube's Common-Music composition language as well as Paul Lansky's RT Mixing program and a special "Emergency Filter" designed just in a couple of minutes by Julius O. Smith.

Pitcture: formal overview of "La cloche sans vallees" with references (algorythms shown

in the score) and transpositions in halftonesteps



DOPPIO SOLO

Luigi Ceccarelli
Musica Verticale
via Tevere, 15 - 00198 Roma
tel/fax +39 6 8411034

Abstract

Doppio Solo (Double Solo) is a composition for amplified alto saxophonist with sampled bass saxophone and alto saxophone sounds. It was realized in 1993 in the composer's private studio with the collaboration of the saxophonist, Federico Mondelci.

The basic idea behind the work is to be found in considering the various creative possibilities which can occur when a direct confrontation is set up between a live performer and a computer-controlled performance. The piece, decidedly virtuosic in character, demonstrates both the soloist's bravura together with the highest possible level of studio editing techniques. Rather than compromising these two different worlds of techniques the end result yields a true expansion of sonic events. The live and synthetic parts are almost always present together and often strictly intertwined, whether in melodic and rhythmic figures or with timbre or range. Thus the title *Doppio Solo*.

The main technical challenge in the studio was to reconstruct a synthetic part which would match the saxophone's natural phrase characteristics and at the same time create a part which amply surpasses the human limits of possible saxophone technique. Almost all of the individual sounds were sampled separately in order to allow for complete control in freely reconstructing, using digital editing, authentic sounding melodic figures and rhythmic articulation. Only in a few instances were the almost 400 original samples altered electronically. The samples were grouped into five timbral categories: key clicks, slap tongues, staccatos, tremolos, repeated notes and multiphonics. These five groups were in turn further subdivided into 32 different programs and corresponding MIDI channels. Each of the programs consists of a unique timbre for the full range of the instrument. The score and sequence of sampled sounds was produced using a commercial sequencing program on an Atari computer.

....comunque il significato di "Doppio Solo" non va ricercato nei suoni in sé, ma nella relazione che i suoni hanno con la personale storia di ogni ascoltatore.

"Doppio Solo" è una composizione per sax contralto amplificato e suoni campionati di sax basso e di sax contralto. È stata realizzata nel 1993 presso lo studio dell'autore dopo un periodo di ricerca sulla tecnica esecutiva del saxofono compiuta con la collaborazione con Federico Mondelci.

L'idea fondamentale che sta alla base di questo pezzo consiste nel mettere a diretto confronto, nella realizzazione di una stessa idea musicale, un esecutore dal vivo ed una esecuzione realizzata tramite computer, e di considerare come elemento creativo le possibili differenze tra le due.

L'intenzione del compositore è stata quella di realizzare un'opera in cui potessero convivere al massimo livello tecnico ed espressivo le due componenti (quella virtuosistica dell'esecutore e quella, per certi versi altrettanto virtuosistica, del montaggio in studio) ottenendo un risultato che non fosse un compromesso tra tecniche diverse ma una vera e propria espansione degli universi sonori possibili.

La composizione è pensata come un pezzo teso a sfruttare ai

massimi livelli l'abilità del solista. La parte dal vivo e la parte sintetica sono entrambe sempre presenti e spesso sono strettamente unite (ognuna come un'ombra dell'altra), sia come carattere delle figurazioni melodico-ritmiche, che come timbro e tessitura. In realtà abbiamo quindi un solista reale ed uno virtuale che eseguono un medesimo solo. Da qui l'origine del titolo.

In "Doppio Solo" non sono presenti soltanto tecniche diverse, ma anche gli elementi stilistici sono difficilmente riconducibili ad un unico modello precostituito storicamente. Pur mantenendo una innegabile unitarietà, si è cercato infatti di ottenere un risultato musicale al di sopra della classificazione dei generi e delle varie correnti, utilizzando stilemi che nonostante oltre cinquant'anni di convivenza appartengono ancora a concezioni lontane tra loro.

Descrizione della forma.

Il pezzo è diviso in cinque sezioni secondo lo schema A,B,A',C,A". Le sezioni A sono costituite da più linee melodiche molto strette tra loro (una dal vivo ed altre sintetiche in numero variabile da una a cinque) che seguono esattamente lo stesso andamento sinuoso. Le tre sezioni sono differenziate soprattutto dai diversi registri in cui suona il sax contralto: la prima parte è scritta prevalentemente nel registro grave, la seconda

nel registro medio, mentre nel finale del pezzo si raggiunge il registro acutissimo.

Nelle sezioni B e C il rapporto tra suono dal vivo e suono sintetico è meno stretto. La sezione B ha il carattere di fascia sonora continua ed è realizzata unicamente da sovrapposizioni di tremoli, ribattuti soffiati e colpi di chiave che formano una denso agglomerato armonico cangiante. In questa sezione il solista ha una parte di tremoli molto libera.

La sezione C, all'opposto, è realizzata solo con suoni cortissimi. La parte sintetica è composta unicamente da un tessuto ritmico di *slap* (colpi d'ancia) del sax basso: fino a otto linee ritmiche semplici, di poco sfasate tra loro per una diversa velocità metronomica, sono state sovrapposte in modo da formare un pattern ritmico molto complesso e variabile gradualmente.

La durata totale del pezzo è di 13 minuti.

Realizzazione tecnica.

Il problema principale affrontato nella realizzazione in studio della parte sintetica è stato quello della ricostruzione del carattere saxofonistico delle frasi che fosse nello stesso tempo credibile ma che superasse ampiamente le possibilità tecniche dello strumentista.

Questo risultato è stato ottenuto in primo luogo grazie ad un lavoro che in ogni sua fase (registrazione, organizzazione

dei campioni e realizzazione delle sequenze) ha sempre mantenuto un alto grado di qualità del suono.

Per la registrazione digitale dei suoni dei saxofoni di Federico Mondelci sono stati utilizzati due microfoni Neuman U87 posti molto vicini agli strumenti in modo da registrare anche i rumori della meccanica ed il soffio dell'esecutore, componenti che in questo pezzo hanno un ruolo non trascurabile. Dalla registrazione sono stati in seguito selezionati e ricampionati circa 400 diversi suoni che sono poi stati memorizzati con una frequenza di campionamento di 44.1 KHz in due campionatori Akai S1000 aventi una capacità totale di circa 18Mbyte. Il fatto di usare due campionatori non è tanto legato ad un problema di capacità della memoria, ma alla possibilità di avere contemporaneamente 32 suoni invece di 16 e di potere organizzare 32 programmi completamente indipendenti dal punto di vista della comunicazione via MIDI.

Per potere ricostruire liberamente con il montaggio digitale l'articolazione melodico-ritmica i suoni sono stati campionati tutti singolarmente (fanno eccezione alcuni tremoli, alcuni ribattuti ed alcuni suoni multifonici tenuti che hanno la durata di qualche secondo). Questa scelta è stata fatta per potere controllare completamente la ricostruzione delle frasi musicali e, pur cercando di mantenere sempre una

stretta analogia con la qualità espressiva dell'esecuzione dal vivo, superare i limiti tecnici degli strumenti meccanici, soprattutto per la velocità e la chiarezza dell'articolazione.

L'organizzazione dei campioni è stata fatta secondo cinque diverse tipologie timbriche: suoni di chiavi, slap, staccati, tremoli, ribattuti e multifonici. Inoltre i campioni sono stati suddivisi in 32 programmi diversi assegnati ad altrettanti canali MIDI. In ogni programma è contenuto un diverso timbro che riproduce tutta l'estensione dello strumento reale.

Generalmente si è cercato di mantenere il più possibile la riconoscibilità del suono originale senza alterare elettronicamente il segnale digitale. Soltanto alcuni tremoli sono stati abbassati di una o più ottave per ottenere una funzione di bordone. Anche alcuni slap, in una breve parte del pezzo, sono stati raddoppiati di velocità e quindi alzati di una ottava, per ottenere un suono non realistico.

Il progetto della partitura e della sequenza dei campioni è stato realizzato con un comune sequencer commerciale su computer Atari.

Nell'organizzazione delle sequenze dei suoni, alcune soluzioni adottate sono degne di nota:

- Per le tre sezioni A è stata creata una sequenza indipendente, parallela alle sequenze melodiche, per il rumore delle

chiavi. Questa soluzione, oltre a rendere più reale l'effetto complessivo, ha permesso di accentuare più o meno la precisione dell'articolazione ritmica indipendentemente dagli altri suoni, semplicemente variando l'intensità dei colpi di chiave;

- La velocità metronomica subisce continuamente piccole e quasi impercettibili variazioni che servono ad accentuare il carattere dell'andamento delle frasi, come accade realmente in una esecuzione dal vivo. Generalmente si ha una piccola accelerazione nei crescendo dinamici ed un rallentamento nei decrescendo e nel finale di ogni frase.

FINZIONI FOR VIOLIN AND QUADRAPHONIC TAPE

Fabio Cifariello Ciardi

Conservatorio "U.Giordano"-Foggia
via Pietro Giannone, 28
00195 Rome - ITALY

Abstract

In "Finzioni", for violin and quadraphonic tape [1], I considered listeners' common long-term memory as a pole in the exploration of a sonic universe entirely derived from violin samples.

In music cognition memory is usually divided into short-term (episodic) and long-term (semantic) memory. The first refers to the occurrences of specific, concrete, temporally dated events, the latter concerns the organized, abstract, timeless knowledge a person possesses. The use of short-term memory is illustrated by the recognition of repetitions and variations of an unknown event presented immediately before. The use of long-term memory is illustrated by the correlation of an unknown event with a personal and social experience. This task is always performed during music perception in order to store the overall meaning, to grasp the

pattern invariants of pieces and styles [2],[3]. Compared to short-term memory, long-term memory is rarely considered in compositional processes. It has been often treated as a personal, unique wisdom that loses its consistency when applied to a large group of people. Furthermore long-term memory refers to knowledge that is already achieved while composers ideally tend to explore original, unknown territories. On the other hand common auditory experiences today are much more frequent than in the past and therefore we might expect to store the same experiences in a much more tangible common long-term memory. This common knowledge play an essential role in timbral perception of natural and artificial sounds. Long-term memory knowledge is retrieved in order to synthesize timbre multidimensionality into a more invariable "mental image", no matter whether the sound comes directly from real instruments or from loudspeakers.

According to contemporary research in music cognition these structural constancies underlying surface changes (like spectral variations in time domain) could be related to archetypical physical processes we associate to sound, such as plucking, striking, bowing, blowing [4],[5].

These reflections have been taken into account in the course of the timbric organization of "Finzioni". All sounds have been derived from real violin samples only, but their modifications have been carried out to create a dialectic interplay between musical events and the common memory of the sonic universe the violin is able to evoke.

I considered our common mental "image" of the violin as the result of a complex interaction among three different memories: a timbric memory, a memory of instrumental gestures and a memory of very well-known musical fragments. In this sense the exploration of the violin's sonic universe in the tape part is intended as the exploration of an interplay between the music and these three memories.

In the tape part, timbric memory is explored through a slow and irregular interpolation going from the deeply transformed sound of the beginning which prolongs tone ending the solo introduction like a frozen resonance, to the pure violin tones at the end which recall some figures previously played by the soloist.

The interplay with our memory of instrumental gesture is stimulated by artificial granular articulations of transformed violin samples, but also by "untouched" samples of unorthodox violin articulations (like the smooth passage between normal and harmonic tones in the central part of the piece) embedded between natural violin samples articulated in an artificial way.

Finally, fragments from a very well known violin repertoire have been used to interact with a common musical memory. These fragments are:

- 1) the first theme from the "Danse Macabre" by Camille Saint-Saens;
- 2) the first theme from "The Sorcerer's Apprentice" by Paul Dukas;
- 3) The first theme of the third movements of Brahms's violin concerto;
- 4) The first theme of the third movement of Brahms'violin sonata in D minor op.108;
- 5) The evergreen "The man I love" by George Gershwin.

All fragments join the same melodic micro-structure that is used as basic material for the all pitch structure of the piece.

"Finzioni" is the title of an anthology of short stories by Jorge Luis Borges. Usually Borges astonishes the reader with his associations between real and imaginary worlds, between things already stored in our common memory and things

which are totally new. In this same sense I imagined this music travelling quickly, sharply through the virtual space of our memory, a virtual space simulated by the quadraphonic reproduction system. The soloist should be placed at center of this space, not on the stage: the entire sonic universe is dreamed of, created and explored by him. A soloist who, at the end of the "Fiction", will pass, like Alice, ... through the looking glass.

Borges described all that in "The circular ruins", one of "Finzioni" novels:

"Nessuno lo vide sbarcare nella notte unanime..."

...

Il proposito che lo guidava non era impossibile anche se soprannaturale. Voleva sognare un altro uomo: voleva sognarlo con minuziosa interezza e imporlo alla realtà...

...

Lo sogno attivo, caldo, segreto...

...

Gradualmente lo venne avvezzando alla realtà...

...

In un alba senza uccelli il mago vide avventarsi contro le sue mura l'incendio concentrico...

...

Andò incontro ai gironi di fuoco: che non morsero la sua carne, che lo accarezzarono e inondarono senza calore e senza combustione. Con sollievo, con umiliazione, con terrore, comprese che era anche lui una

parvenza, che un altro stava sognando.

"Finzioni" has been released at IRCAM during the Cursus d'Informatique Musicale 1991 and is dedicated to my son Francesco.

[1] F.Cifariello Ciardi: "Finzioni" per violino e nastro qudrifonico", EDIPAN ed. Roma, 1992.

[2] W.J.Dowling & D.L.Harwood: "Music cognition", Academic Press, New York, 1986.

[3] A.Baddeley: "La memoria umana" Il Mulino ed.,Bologna, 1990.

[4] C.Cado: "Timbre et causalité" in "Le timbre, métaphore pour la composition", IRCA M-C.Bourgois ed., Parigi, 1991.

[5] G.J.Balzano: "What are musical pitch and timbre", Music Perception, Vol.3,N.3, pp.297-314, 1986.

VISIBILI FOR TWO VIOLINS AND TAPE

Alessandro Cipriani

Via Voghera, 7 00182 Roma Tel.++39 6 7010310

"Visibili" was composed in 1992, a period of the history of music when the passage from the techniques of written tradition to those of electronic tradition becomes more and more evident: if we think about the influence of new technologies in recording studios on the performance and the recording of classical and contemporary music; if we notice the fact that listening to music in our society occurs mostly via technology (CD, tape recorders, media, etc.) and not directly; if we think about the impact that the proliferation of music everywhere has on the value and the meaning of the performance and of the listening ritual, we can become aware of the importance of that passage.

This piece was born from research on the possibility of using the same basic instrumental materials (written or played and recorded) and two different elaboration techniques: the first, bound to the written tradition, for the instrumental parts; the second, bound to the electronic tradition for the elaboration of the tape part. Exchanges among techniques of different traditions constantly took place in the history of music (for example

the survival of numerous signs of oral tradition techniques within the written tradition) [1]. Therefore I tried not to establish theoretically strict boundaries between strategies and behaviours definable as "electronic" and those definable as "instrumental". I tried instead, within the possibilities at hand, to make the best of the specific quality of the two traditions and of the possibilities that the respective techniques offer. The differences among those two "worlds" find their unity, besides that of the basic materials, also in the construction of the form. There I tried to identify some basic formal functions that could be dealt with by one or the other medium: the formal functions and the sound of the instruments as unifying elements; the diversity of the traditions and of the techniques as a subject to reflect on.

The three techniques used here (reiteration, multi-track recording and related techniques, granular synthesis) are indicated in the score by three different symbols. I didn't try to 'describe the sound' since that description would be extremely reductive. This is, for example, a specific

quality of the electronic tradition: the separation of the sound description from its performance.

The first section of the tape of "Visibili" is made accelerating the reiteration of pizzicato sounds up to a speed of reiteration of 100 impulses per second, so that two textures are produced: one has the characteristic taken from the pizzicato (same pitches as the pizzicato sounds performed at the beginning of the piece by the violinists); the other texture is generated by the frequency of the impulses up to 100 Hz. The higher or lower frequency of the reiteration is used as a means to a higher or lower tension on the perceptive level. The idea of the reiteration technique comes from the Stockhausen's proposal [2] of a sound generated by the frequency of impulses of another sound. The difference from that technique is that in "Visibili" I used instrumental sounds that are recognizable by their timbre also when they become a texture, notwithstanding the intervention of the second sound used here with a sort of 'subharmonic' function. This is another of many examples taken from the electronic tradition: to repeat the same sound (and not to perform the same note) is one of the specific possibilities of those means that, since many years, have allowed a sound not to exist only in one moment and in one

place, 'disconnecting' in a way the sound from its source.

I tried, in the second section of the tape of "Visibili" to bring out some aspects of the violin timbre, by first recording an instrumental cell performed by one violinist in different versions (each beginning from a different pitch keeping the intervals unchanged), then "transposing" each of those versions with electronic means to the same beginning pitch some octaves under the original. Comparing the various versions it was easy to notice the diversities within the timbre of the different "transpositions", and yet 'the same notes were played' and all of them were originally played on the same instrument. That is another specific technique of the electronic tradition: not to play a lower pitch note, which will have necessarily a different harmonic spectrum configuration from a higher pitch note played on the same instrument, but 'to transpose' the same note (keeping the same relation among the harmonics) in a lower register. The latter way the production of a certain type of harmonic spectrum is 'disconnected' from the typical pitches to which the harmonic spectrum is connected. To use a type of 'electronic' technique means in this case to conduct research at the same time into the sound of the instruments, the

same ones that will be played live.

The third technique is granular synthesis and, in particular, the technique developed by Barry Truax to 'time-stretch' a sound without changing its pitch. In this case, instead of using a real-time system such as Truax's PODX, I used in the penultimate and more extensively in the last section of the piece a CSound orchestra written by Eugenio Giordani.

"Visibili" was also born from reflection on the concreteness, the physical quality, the visibility of the performer, on his relation with the instrument (with all the historical connotations implied) on one hand and the 'invisibility' and the sense of abstractness of the 'resounding' of the tape, notwithstanding the concrete original source (violin) on the other. I will try to describe the scene of the performance: the two violinists play live, one in front of the other with the music stands between them. In the piece there are two sections for tape only: during the first the performers stay motionless, one in front of the other seen sideways by the public; before the second section they leave the stage walking towards the wings and playing the last notes prescribed in the score. At this point on the stage, behind the music stands appears a motionless natural-size image of the two violinists wearing the

same clothes, in the same position where the real performers were formerly playing. The projection takes place using a slide projector but the image has been previously photographed from a video so that the texture of the human figure refers more to an electronic image rather than to a photo. At the end of that last section for tape only a two-note chord can be heard from afar: it's the sound of the two violinists who are behind the wings...

References

- [1] L. Treitler "*From ritual through language to music*", Schweizer Jahrbuch fuer Musikwissenschaft, Neue Folge 2, pp. 109-123, 1982.
- [2] K. Stockhausen "*L'unità del tempo musicale*" in 'La musica elettronica' edited by H. Pousseur, Feltrinelli, pp. 150-160, 1976 (also in german language) in Zeugnisse, Europaeische Verlaganstalt, Frankfurt, 1963.
- [3] A. Cipriani "*Verso una tradizione elettronica?*" unpublished, 1993.

RECONSTRUCTIONS FOR HARP AND COMPUTER

James Dashow

02030 Poggio S. Lorenzo (RI)
FAX: (+39) - 765 - 80.294
or (+39) - 6 - 58.19.904

The organizers of this Colloquium have asked us composers to say a few words about our non-wordy work. They have requested us, kindly, to attempt to provide a few clues about the more significant musical or technical or, if all else fails, methodological characteristics of our efforts, in a manner (hopefully) comprehensible (emphasized by italics) to readers outside our specialty (just about everybody). So, dear reader, I will herein endeavor to do all as asked by the hard-working organizers of the X Colloquium in the interests of comprehensibility, and in accordance with the "Instructions for Composers" provided by them to help us non-verbal types navigate the difficulties of expressing ourselves in an unfamiliar medium.

Of course, all the words in the world won't help you get through RECONSTRUCTIONS if you don't have a chance to hear it. So those readers who were not at the 4 December concert in Milano (or even those

who were but would like a few more goes at it), should head for their favorite local CD store and ask about a ProViva CD, made in Germany and coming out in early 1994, with 4 of my pieces on it. RECONSTRUCTIONS is one of them, and the harpist for whom it was written, Lucia Bova, is the soloist. She plays it extremely well. Then you should listen to it, ears wide open, and first of all just enjoy (or get accustomed to) the sound.

After you have listened once, maybe twice, you will have undoubtedly noticed that it is asymmetrically in two parts: the first part energetic, multi-dimensional linear and full of complexities, the second much slower and suspended, more concerned with developing sound colorings rather than any other aspect. But the first part has a slow suspended pause in it that is a forward reference to what is coming, while the second part has a highly charged energetic part in it which is a backward reference to what has been. The combinations of intervals and the specific pitches that make up

each section and provide motivations for the forward/backward references are Significant Musical Characteristics. But if you don't happen to have perfect pitch or are unable to memorize interval patterns the first few times around, don't worry. The form of the piece is also, and especially, concerned with kinds and degrees of energy and motion, and this is all right out there in the open for you to hear. All those pitch and interval relationships support (and are supported by) the flux and transformations of energy of the piece. My job is, has been, to convince you that it all goes together and makes musical sense, as it does to me.

Would my telling you a few of my workshop secrets be of any value to you, dear reader (and now listener, I hope)? Well, maybe. You might have noticed that the harp pitches are often there in the electronic sounds. Well, actually the harp notes are always (*italic* that one) there in the electronic sounds, sometimes in the foreground, sometimes in the background. Yes, dear reader-listener, there are indeed some specific sound synthesis algorithms involved, all written by yours truly. I can make the computer generate all sorts of sounds built around whatever pairs of notes (intervals) I choose. All the sounds are structurally derived from the

pitches played by the harp, and vice versa, all the pitches are organically related to the electronic sounds, or timbres. Pitch structure and timbre structure are completely integrated. Or more simply, the harp and the electronic sounds are one and the same structure - each part works out, within, around the structure in its own way. Kind of like simultaneous (rather than successive) variations on the same musical stuff.

So, I think up ideas based on some of those rich electronic sounds, or on some combination (successive or simultaneous) of those intervals, or interactions between both, and then I elaborate them with all kinds of Significant Methodologies. I even have a method for the development of the underlying pitch structure which in turn is made to generate the sounds. If you want to know more about the nuts and bolts of these methods you can look at some of my other attempts at prose, printed, more or less comprehensibly, elsewhere: [1] [2] [3] [4] [5].

But to complete the picture with some Technique: the electronic sounds were entirely generated using the MUSIC30 software system for digital sound synthesis. MUSIC30 was written by myself for the PC-compatible plug-in accelerator board known as the SPIRIT-30, made by some very cooperative electronic

engineers at Sonitech International [6] utilizing the Texas Instruments DSP chip, the TMS320C30. For those of you who are concerned: the system is so fast that many things can be done in real time, while deferred time performance of complex music is close to mainframe standards. All with a humble PC. You can work at home.

Where does the title of this piece come from? A nice quote from Jean Piaget, which goes like this: "Reconstruction involves increasingly varied recombinations and a greater freedom in the kinds of combinations."

In fact, Piaget has an interesting, different, (musically applicable) way of thinking about structure to which I subscribe, and which I think is worthwhile mentioning here too: "...structure is a system of transformations. Inasmuch as it is a system and not a mere collection of elements and their properties, these transformations involve laws: the structure is preserved or enriched by the interplay of its transformation laws...the notion of structure is comprised of three key ideas: the idea of wholeness, the idea of transformation and the idea of self-regulation."

That pretty well describes the way I composed this piece, and others too.

But then there is the usual question, what does it all mean.

Well, all you have to do is listen and find out. Buon ascolto.

References

- [1] James Dashow: interview with S. Momo in booklet accompanying CD recording of 3 works (ARCHIMEDES, Act I, scene ii; MNEMONICS for violin and computer; ORO, ARGENTO & LEGNO for flute and computer), WERGO CD WER 2018-50, 1989.
- [2] James Dashow: "*Spectra as Chords*", Computer Music Journal, Vol. 4 N. 1 (Spring), pp. 43-52, 1980.
- [3] James Dashow: "*Three Methods for the Digital Synthesis of Chordal Structures with Non-Harmonic Partialis*", Interface, N.7 pp. 69-94, 1978.
- [4] James Dashow: "*New Approaches to Digital Sound Synthesis and Transformation*", Computer Music Journal, Vol. 10, N. 4 (Winter), 1986.
- [5] James Dashow: "*The Dyad System*", Perspectives of New Music, forthcoming.
- [6] Brewster LaMacchia, Yogendra Jain, Sonitech International, 14 Mica Lane, Wellesley, MA 02181, USA; tell 'em I sent you.

Sulla composizione di *Zeitwerk (l'orizzonte delle cose)*

4-track computer-generated tape

Agostino Di Scipio

Abstract

The realization of *Zeitwerk (l'orizzonte delle cose)* involved 4*4 processes of timbral exploration (four processes replicated on 4 channels). Each process utilized a unique set of 8 purely sinusoidal grains of fixed frequency and amplitude. Therefore the strategy determined exclusively how the grains repeat and overlap through time (in this sense, the entire composition was realized using design processes operating in the time domain *only*). Criteria of patterning of grains included iterated difference equations showing global properties of self-organisation, ranging from chaotic behaviors to hierarchically structured patterns. Also iterated *time-shifting* and *-folding* of sound were extensively used.

In this short paper, *Zeitwerk* is described as an example of algorithmic composition that brings forth relevant morphological properties of sound materials. In terms of *composition-theory*, the approach is one that merges *models of materials* and *models of musical design*, and that fosters a peculiar attitude towards *indeterministic models of detailed sonic design*.

Composed at the University of Padova with recently implemented granular synthesis options in G.Tisato's ICMS, the work was premiered in Rome (September 1992). The version presented at the 10th CIM was realized later and initially presented at City University, London (April 1993).

1.Introduzione

Il brano si compone di quattro sezioni caratterizzate da specifiche forme timbriche. Ogni sezione è realizzata attraverso quattro processi simultanei di trasformazione timbrica (uno per canale). Alla base della composizione, vi è un'unica strategia di progettazione non deterministica del suono, che

utilizza null'altro che 8 minuscoli grani sonori, di frequenza ed ampiezza fissa; di tali grani vengono determinati dunque solo la durata ed i modi di successione e sovrapposizione nel tempo. Perciò la costruzione dell'intero brano è avvenuta lavorando *solo* nel dominio del tempo (come allude il titolo, che però in tedesco ha anche connotazione

fortemente autoironica, significando infatti "opera di rilevanza epocale"...). Pur fisso per tutto il brano, le frequenze degli 8 grani hanno valori strategicamente importanti, ovvero : 48, 105, 232, 511, 1124, 2473, 5442, 11972 Hz. Spettri con simile contenuto di frequenza inducono illusioni di percezione di altezza. Ho così potuto raddoppiare o dimezzare la frequenza di campionamento per utilizzare più di una volta alcune strutture sonore, la modifica dando luogo alla percezione di un semitono di differenza, invece che di un'ottava. (Naturalmente, il sovra-campionamento ha anche dato luogo a fenomeni, ben accetti, di *foldover*).

2.Processi microtemporali

I controlli che determinano le relazioni temporali tra i grani sonori sono stati derivati da modelli di sistemi dinamici non-lineari (iterazione di equazioni-differenza del primo ordine), il cui utilizzo come struttura di controllo della sintesi granulare ho discusso altrove [1,2]. Nell'evoluzione di un sistema non-lineare, l'errore di predizione cresce logaritmicamente, così che diventa impossibile anticiparne il comportamento a seguito di stati iniziali quasi identici (*sensibilità alle condizioni iniziali*). Le proprietà di auto-organizzazione di tali processi nonlineari, proiettate nella microstruttura del suono, hanno fatto emergere morfologie timbriche di

vario tipo, anche molto differenti tra loro.

Su alcune delle strutture sonore così generate, sono state applicate anche operazioni di *time-shifting* del suono (dilatazione temporale senza alterazione di frequenza) e di *foldover temporale* (due strategie di *time-folding* : [lineare] stabiliscono una soglia di durata massima, i grani che la oltrepassano vengono ribaltati e rimixati con i campioni già generati; [nonlineare] ricampionamento di un file sonoro esistente mediante puntatore governato da *brownian walk*, o da una iterazione non-lineare). Tutte le operazioni sono state applicate ricorsivamente, sottponendo a granulazione suoni costituiti da innumerevoli grani. Se effettuata in modo appropriato, la rigranulazione allarga le bande di frequenza del suono che viene elaborato, provocando temporanee discontinuità nel segnale e dunque suoni estremamente ricchi e complessi, fino al rumore.

3.Teatro. Composizione algoritmica e *timbre composition*

Come per altri miei recenti lavori, anche *Zeitwerk* è un esempio di composizione algoritmica operata al livello della struttura dei materiali sonori e, in definitiva, del timbro. In altre parole, si rendono inseparabili pensiero dei materiali e pensiero della forma, ovvero, si *confondono* modelli di

generazione del materiale sonoro e modelli della articolazione della forma musicale [3]. Da questa fusione, che diventa praticabile attraverso l'utilizzo di rappresentazioni microstrutturali del suono (in questo caso sintesi granulare, ma altri sono possibili), nascono modelli *indeterministici* di progettazione del materiale sonoro : comporre il timbro come strategia di esplorazione e scoperta.

Ciascuna delle quattro sezioni del lavoro si sviluppa fintanto che il processo di sintesi del suono e l'evoluzione della materia sonora che ne deriva si traducono in mutazioni ed insorgenze timbriche di interesse percettivo. Oltre l'orizzonte di tempo entro il quale la microstruttura si rigenera in ulteriori *forme possibili del suono*, le sonorità si cristallizzano, si stabilizzano; ed a quel punto viene dato inizio alla sezione successiva del brano. I momenti terminali di ogni sezione non sono da considerarsi punti di arrivo, obiettivi di un discorso mirato, direzionato - ma ipotesi di cammino, punti di temporaneo equilibrio il cui raggiungimento passa attraverso stati morfologici singolarmente caratterizzati ed a volte centrifughi ed in contrasto con la consistenza formale dell'intero. (Credo che *comporre il timbro* conduca potenzialmente ad una esperienza compositiva ed ascoltativa che poco ha della articolazione sintattica, logica e

lineare, che molta musica elettroacustica oggi ancora mutua dai paradigmi del linguaggio verbale e dalle forme evolute della tradizione strumentale occidentale. Che la chiarezza formale dell'intero sia così sacrificata alla ricchezza morfologica del materiale sonoro - ovvero alle forme del materiale - non può costituire un reale problema : rendere possibili nuovi modi del comporre e dell'ascolto, cioè coerenze cognitive inesplorate, significa sospendere principi cognitivi noti; qui sta, in sostanza, il reale contributo di originalità a portata della musica elettroacustica e per computer; cfr [3, 4]).

Nonostante i grani puramente sinusoidali utilizzati, i processi microtemporali di *Zeitwerk* risultano in sonorità talvolta anche molto aspre e concrete, che a tratti si fanno tessiture e rumori di carattere quasi ambientale (a ciò contribuisce la spazializzazione del suono, in termini di ritardo e simultaneità, di densità di eventi sonori simili ma non identici sui 4 altoparlanti). Altre volte, come nella seconda sezione, si hanno sonorità quasi strumentali, ma come di strumenti non riconoscibili, mai uditi.

4.Epilogo

La varietà delle soluzioni timbriche risponde al tentativo di sottrarre la sintesi e dell'elaborazione granulare del suono alle

soluzioni tecniche e percettive a cui è spesso associata. Sono state perciò sviluppate apposite procedure per la sintesi del suono (in tempo *rigorosamente* non-reale...); questi programmi, da me inizialmente scritti su un PC486, sono state poi trasportati e resi più efficienti da G.Tisato sul suo ICMS al Centro di Calcolo di Padova (vedi Di Scipio & Tisato altrove in questo volume e [5]).

Zeitwerk è stato presentato all'Università "La Sapienza" di Roma (Settembre 1992). La versione odierna è stata messa a punto in seguito, ed è stata presentata alla City University (Londra, Aprile 1993).

Riferimenti bibliografici

[1] A.Di Scipio: "*Composition by exploration of non-linear dynamical systems*", Proc. of the ICMC90, ICMA, pp.324-329, 1990

[2] A. Di Scipio: "*Caos deterministico, composizione e sintesi del suono*", Atti del IX CIM, AIMI/DIST, pp.337-350, 1991

[3] A.Di Scipio: "*Models of materials and of musical design become inseparable. A study in composition-theory*", Proc. of the International Conference on Cognitive Musicology, Univ. of Jyväskylä, pp.300-316, 1993

[4] A.Di Scipio: "*La musica di due culture. Tracce di una mutazione*". In (a cura di C.Boschi) *Musica/Scienza. Il margine sottile*, ISMEZ, pp.17-70, 1991

[5] A.Di Scipio: "*Composing with granular synthesis and the Interactive Computer Music System*", Proc. of the 4th Symposium on Arts & Technology, Connecticut College, pp.38-52, 1993

Agostino Di Scipio si è diplomato in Composizione e Musica Elettronica presso il Conservatorio di L'Aquila, dove ha studiato con G.Bizzi, M.Cardi e M.Lupone. Ha inoltre studiato con J.Dashow al CSC dell'Università di Padova.

Collaboratore del CSC, Padova, e del CRM, Roma. Membro fondatore dell'ESCOM (*European Society for the Cognitive Sciences of Music*). Sue musiche sono state eseguite in vari festival in Italia ed all'estero (Roma, Padova, Cagliari, Bologna, Amsterdam, Ghent, Lyon, Bourges, Vancouver, Londra, Norimberga). Inoltre ha presentato lavori di ricerca in molteplici convegni internazionali (Glasgow, Montreal, Londra, Trieste, Roma, New London, Cambridge, Jyväskylä, S.Josè, Genova, ecc...). Menzione di merito a Bourges 1991; nel 1993 ha lavorato presso la Simon Fraser University (Burnaby, Canada) con una borsa dell'*International Council for Canadian Studies*.

ANIMALI IN SOFFITTA

Amedeo Gaggiolo, Silvia Dini

Jour d'atelier
Salita Brasile 13/37 Dx
16162 Genova (Italy)
fax +39 10 310466
phone. +39 10 404419
E-mail musica@ice.ge.cnr.it

The composition “Animali in Soffitta” (Animals in the Attic) consists of four musical episodes intersected by a vocal ‘parenthesis’ deriving from the processing of a sampled female voice.

The piece draws its inspiration from recesses of the mind that swarm with fantastic creatures and which only our centuries-old imagination can digitize in the memory and bring to life.

It is a musical portrait of unreal animals, beasts that for generations have fuelled peoples’ fears, anguish and curiosity.

These animals represent an ancestral presence that roams the mind without respite, taking on multiform shapes; they are incomplete beings denied the possibility of death, a privilege that only the living are granted.

Such is the tale of “A Bao A Qu”, from an oriental legend [1], whose unquenchable thirst for ascent and

purification is embodied by a hopeless dynamic loop.

Another creature is the “Mirmicoleone”, defined by Flaubert [2] as “a lion at the front and an ant at the back”: a biological impossibility and as such destined to perish.

Misery and sadness also mark the story of the “Squonk” [3], an ever-weeping twilight creature thought to live in Pennsylvania; its fear of hunters is so overwhelming that when the animal feels cornered it resorts to “dissolving into tears”.

The “Odradek”, a small sinewy beast springing from the imagination of Kafka [4], is an incredibly fast creature which wanders about people’s homes, agile and uncatchable.

It is described as being “able to live in attics, under stairs, in corridors, entrance-halls.... It may move into the neighbouring houses but it always returns to ours”.

Musical characteristics

Our composition falls within a tradition of “animals in music” which has intrigued musicians of all periods.

What relationship may arise between music and animals? Musical literature provides all kinds of approaches to animal representation and “Animali in soffitta” takes into account the descriptive techniques forming the traditional legacy; the animals, as introduced in Borges’s “Manual of Fantastic Zoology”[5], have been musically interpreted in a ‘phonosymbolic’ way (onomatopoeia, kinetic illusions, echoic symbolism, etc.) [6].

In “A Bao A Qu” the dominant feature is rhythm (fig. 1-2) or, rather, a kind of percussive stammering in which a gong evokes the Orient and marks a ritual stage within the search for purification leading to the summit of the Victory Tower.

The Mirmicoleone’s double nature is conveyed by a melody writhing between low and high pitches, immersed in a hazy landscape of sand and dust produced by pseudo-maracas (fig. 3).

A synthetic polyphonic wailing and a trickle of percussion describe the Squonk’s wretched sobbing and wandering (fig. 4).

The Odradek’s frantic groove recalls a labyrinthian video game, or a tap dance à la Fred Astaire,

interposed with puffs, breathing, exclamations and spurts which, considering the animal’s nature, can only be metallic in timbre (fig. 5).

“Rafel mai amech zabi almi” [7] is Dante’s linguistic enigma, which somehow enwraps all four fantastic animals. The indecipherable voice is the attic - it does not reveal anything of the creatures it houses but echoes the darkness and abyss of being (fig. 6).

References

- [1] A. Cesaro, C. Pansera, U. Rizzitano, V. Vacca ed.: *“Le mille e una notte”*, Einaudi, 1975.
- [2] G. Flaubert: *“La tentazione di Sant’Antonio”*, in *“Romanzi e racconti”*, Mursia, 1961.
- [3] Cox W. T.: *“Fearsome creatures of the lumberwoods”*, Washington Press, 1910.
- [4] F. Kafka: *“Tutti i racconti”*, Arnoldo Mondadori, 1979.
- [5] J. L. Borges: *“Manuale di zoologia fantastica”*, Einaudi, 1962.
- [6] C. Cano: *“Simboli sonori”*, Franco Angeli, 1985.
- [7] D. Alighieri: *“La Divina Commedia”*, *Inferno*, XXXI, 67, La Nuova Italia, 1968.

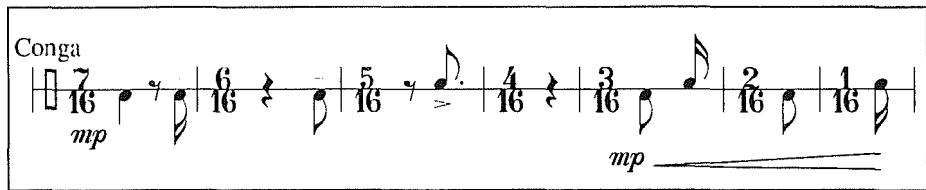


Fig. 1 A Bao A Qu's steps beat to the rhythm of the percussion.

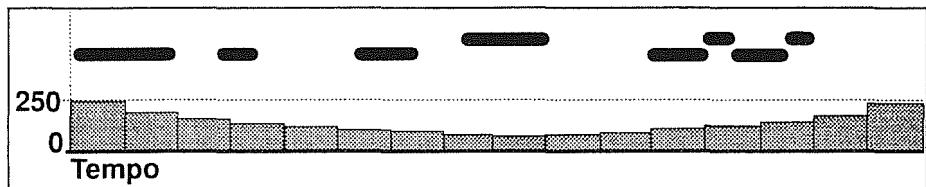


Fig. 2 Representation of A Bao A Qu's swaying motion.

$\text{d} = 48$

Synth

$2:\frac{7}{4}$

pp <> <>

Maracas (Samples)

$\frac{7}{4}$ *p*

p *pp*

Synth

$2:\frac{7}{4}$

pp

Fig. 3 The Mirmicoleone's dual melodic anatomy.

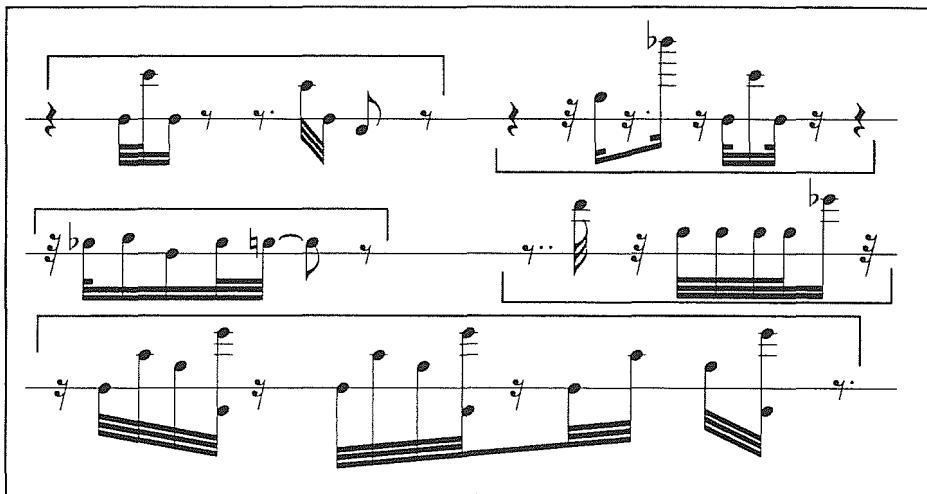


Fig. 4 A few strokes on the synthesizer describe the Squonk's tears.

Tip-tap groove

Metal Percussions

Breath (sample)

Fig. 5 The sampled tap-dance groove adds ironic colour to the Odradek's existence.

Fig. 6 Melodic frame confining Dante's phrase.

M E T A F O N I E

Francesco Galante

V. Taranto, 178
00182 Roma
Tel. +39 6 7027370

The title of the piece may suggest several kinds of meaning.

In this case the term of Metafonie is a valid expression to interpret my interest toward some different kind of perception ambiguities, that I used into the last my musical works.

The aim is to conduct the hearing through different changing conditions starting from a collection of simple sound objects. Consequently the central point of the piece is to obtain different and contrasting perceptive phenomena, colours, ambiguous soundscapes.

I used heavy and smoothed sounds, simple intervals and sound "clots", dramatics and ecstatic character, electronic sounds and electronic sounds that simulate instrumental timbres, and so on.

In particular these virtual instruments are synthesized by using FM and then processed by using of classic techniques of electroacoustic music such as Amplitude and Ring Modulation, Delay, Reverberation, etc.

Metafonie is a tape only music piece and it has been realized at author's studio in Rome, in the first quarter of the 1993, by using 2 FM expander TX81z YAMAHA controlled by Atari 1040 PC, a TMS320 DSP Audio board controlled by 286PC and DSP SPX90-II YAMAHA.

The software is based on a set of customized routines able to generate pseudo-random micro-structures that I used like objects that I can manipulate by Cubase Midi software. For TMS320 DSP board I used customized TI320 assembler code programs.

Metafonie is dedicated to Luigi Pestalozza.

Il titolo può sottoindicare una molteplice quantità di significati, suggeriti dalle sue implicazioni sia in campo linguistico che scientifico.

In questo caso il termine Metafonie ben rappresenta un problema che ha caratterizzato la mia scrittura musicale di questi ultimi anni, vale a dire condurre

l'ascolto verso stati di diversa trasformazione della materia acustica; considerare il movimento all'interno del suono, come aspetto centrale del progetto musicale e sonoro, e quindi della sua forma.

Di conseguenza Metafonie si avvale di quelle tensioni ed instabilità che un materiale, opportunamente progettato, è in grado di produrre nel muoversi da uno status spettrale e percettivo ad un altro, per fissare l'attenzione su istanti di suono, ovvero sulle distanze tra i diversi punti di trasformazione del suono.

Il brano si articola su 3 percorsi che si intrecciano, i quali progressivamente sviluppano le parti "metafoniche" appunto dei materiali utilizzati.

In un movimento sempre più teso ed evidente che conduce l'ascolto da un riconoscibile ambiente sonoro fortemente "strumentale", non soltanto nel senso puramente timbrico ma anche della scrittura, ad un progressivo tentativo di raggiungere spazi acustici inusuali, opposti, contrastati.

METAFONIE è costruito sui contrasti, sul passaggio continuo tra i caratteri contrastanti od ambigui dei materiali.

Suoni grezzi, metallici; uso di registri gravi, acuti ed acutissimi; contrapposizione di semplici relazioni intervallari con "grumi"

fonici; carattere drammatico ed estatico, sintesi del suono elettronico puro e sintesi del suono che tenta una mimesi del suono strumentale.

Tutti questi elementi sono stati presi in considerazione durante la stesura e la realizzazione del progetto.

Gli "strumenti virtuali", suoni di pseudo-ottoni ed archi, prodotti dai sintetizzatori in FM vengono utilizzati con una forte caratterizzazione ambigua, costruendo un intreccio dialogico in cui coesistono figurazioni orchestrali che appartengono a tendenze linguistiche avanzate, così come procedimenti di elaborazione massiva degli strumenti virtuali attraverso processamenti elettronici classici (ringmodulazione, reverberazione).

Un insieme di metafore sonore quindi che in altra maniera traduce il rapporto strumenti concreti/elaborazione elettroacustica tipico di una certa prassi musicale, in quello di strumenti digitali midi/elaborazione elettroacustica.

Nella costruzione del materiale sono stati adottati procedimenti di generazione pseudo-random di microsequenze, in cui sono stati definiti gli ambiti parametrici di accadimento degli eventi.

Le microsequenze sono state memorizzate come tracce midi all'interno di un ambiente di la-

voro CUBASE per Atari, in questo modo esse sono state successivamente trattate ed organizzate come oggetti, rendendo possibile la costruzione nello spazio musicale di tessiture sonore variabili in velocità e collocazione frequenziale, e nella verticalità possibile data dalle potenzialità degli strumenti MIDI impiegati.

Una metodica che ha contribuito fortemente alla sperimentazione di differenti complessità timbriche e dinamiche.

In particolare, si è adottato un criterio di generazione continua a "burst" delle note, in modo tale da ottenere una mutazione dinamica del materiale prodotto dai 2 sintetizzatori digitali, sfruttando sempre il massimo delle risorse disponibili.

Questa tecnica produce all'ascolto dei microistanti di instabilità percettiva, rumori dovuti alla continua sostituzione di un evento con l'altro. Controllando la velocità con cui attivare questo tipo di generazione e il tempo di sostituzione degli eventi ed integrando mediante complessi filtri digitali disponibili in alcuni algoritmi del DSP SPX90, si è potuto costruire degli spettri acustici con diversa distribuzione dell'energia formantica, particolarmente utili nella progettazione del suono pseudo-strumentale.

La tessitura frequenziale è stata centrata su poche altezze, spesso in distanza di banda critica, ed un certo numero di loro microvarianti (fino a 32 cent dal suono base), distribuite su tutte le ottime disponibili per ciascun sintetizzatore digitale e, durante la fase di assemblaggio, su differenti spostamenti di registro.

L'algoritmo di sintesi utilizzato è un modello di strumento FM a 4 operatori, di tipo 4, presente nel TX81Z, opportunamente modificato nei rapporti di frequenza tra gli operatori e nei rispettivi inviluppi.

I materiali di sintesi, così ottenuti vengono successivamente elaborati attraverso una scheda DSP non commerciale, basata sul TMS320, sulla quale si è implementato un programma di simulazione di ringmodulazione e modulazione di ampiezza, con oscillatori ed inviluppi digitali di controllo.

La ricostruzione in chiave audionumerica di questi strumenti di elaborazione tipici del sintetizzatore analogico, hanno favorito un comportamento ed un rapporto con il suono sicuramente molto stimolante, ma ha significato anche progettare un rapporto non convenzionale con gli strumenti midi, che spesso riconducono il compositore al problema di come stravolgere tali mezzi

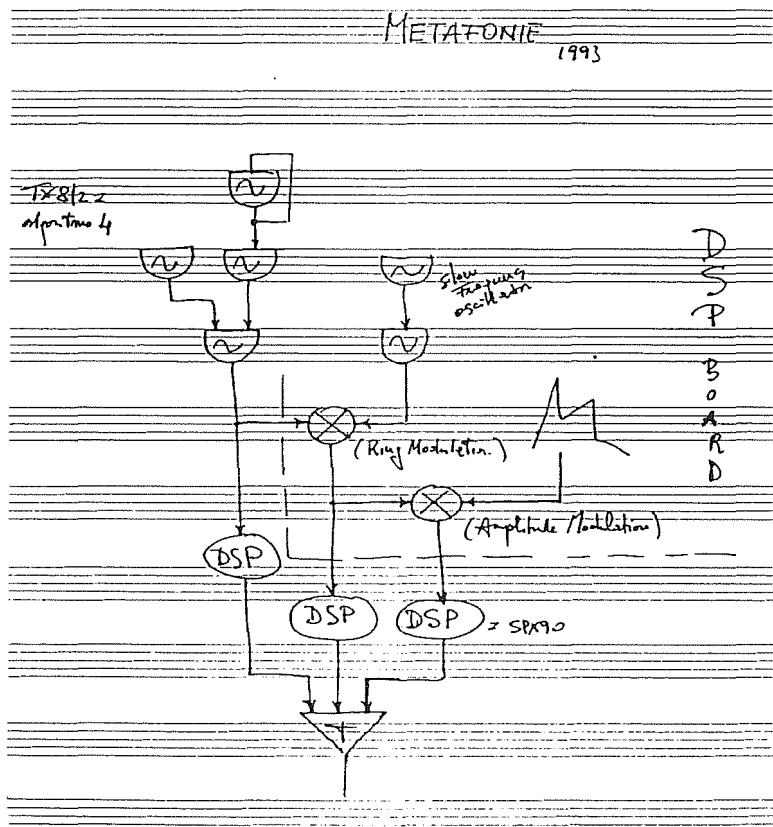
tecnologici, apparentemente "aperti", così come è già avvenuto per gli strumenti tradizionali.

METAFONIE è stato composto nel periodo Gennaio-Aprile 1993 nello studio dell'autore, mediante 2 FM expander TX81Z Yamaha controllati da Atari 1040 pc, una scheda DSP non com-

merciale, basata sul TMS320 (conversione D/A e A/D 16 bit) controllata da un 286PC, un processore digitale di segnali SPX90-2 Yamaha.

La sua esecuzione è esclusivamente per nastro magnetico.

METAFONIE è dedicato a Luigi Pestalozza.



ALCUNE RIFLESSIONI INTORNO AL BRANO ELETTRONICOACUSTICO *CHROMATISM*

Francesco Giomi

Divisione Musicologica CNUCE/C.N.R.
Conservatorio di Musica di Firenze
P.zza Belle Arti, 2
I-50122 Firenze (Italy)
fax: +39-55-2396785
E-mail: art@ifiidg.bitnet

Abstract

The computer music piece *Chromatism* takes as a starting point the author's researches on electroacoustic music analysis, based on the sound object concept. It also integrates some experiences in the field of music and images interaction.

Chromatism includes six studies: at first the studies should have represented six of the twelve colors of the chromatic disk, in order to re-create, at an auditory level, a sort of sound colors. But during its composition the piece has partially lost this component in favour of the creation of completed narrative structures which are developed inside each of the six single fragments. The author tried to link the single narrative paths through an overall structure comprising the six studies.

As far as the sound material is concerned, each fragment takes into consideration particular aspects of the electronic sound world, like the alternation between sound and noise, the rhythm caused by partially random parameters and the environmental characteristics of certain timbral/harmonical

textures. Many of the musical elements are repeated and amplified from one single piece to another in order to create timbral, besides structural, bonds between the six fragments. In the work there are both electronic sounds and sampled acoustic instruments. The first are used in order to emphasize the timbral aspects, trying to insert compound objects characterized by a tonic and/or complex mass. Sampled acoustic instruments were used to create composite events, formed by a rhythmic assemblage of simple objects or by sound "groups" with a rhythmic function.

Chromatism has been realized at the Musicological Department of CNUCE/C.N.R. (Conservatory of Music of Florence) with the automatic composition software Teletau, the Yamaha TX81Z synthesizers and the Roland S550 sampler.

L'art des sons fixés

Un recente libro di Michel Chion [1] ci dà lo spunto, oltre che per il titolo di questo paragrafo, anche per dedicare alcune brevi riflessioni preliminari alla musica elettroacu-

stica per nastro, alla quale il brano appartiene. Il compositore francese pone proprio l'accento sulla validità di una musica progettata, composta e fissata esclusivamente su nastro magnetico. Chion la definisce con il termine "arte dei suoni fissati", ad indicare una musica dove il compositore lavora artigianalmente con i suoni, piuttosto che con i segni, in una costante alternanza tra il fare e l'ascoltare. Suoni che si trasformano gradualmente da materiale a oggetto stesso della composizione, e dove anche l'operazione di fissatura, la registrazione, diventa momento artistico.

Le frequenti obiezioni riguardo alla mancanza del ruolo dell'interprete nella musica elettroacustica possono essere superate, prima di tutto rilevando la presenza di forme di interpretazione anche nei processi percettivi legati all'ascolto [2] e, in secondo luogo, sviluppando, ai fini dell'esecuzione concertistica, nuove forme di diffusione musicale legate soprattutto alla progettazione di specifiche sale di ascolto, strada peraltro già intrapresa da alcuni centri sia europei che nord-americani.

A partire da questi e da altri concetti, quali per esempio quelli illustrati da Pierre Schaeffer già nel 1966 nel suo trattato [3], vengono attualmente condotte numerose ricerche sull'analisi della musica elettroacustica. Queste cercano di formulare criteri per l'individuazione e la classificazione degli oggetti sonori presenti in un brano oltre che per la scoperta delle loro modalità di organizzazione formale e delle loro funzioni estetiche.

Il lavoro musicale *Chromatism* trae origine proprio da alcune di queste ricerche condotte dall'autore [4] e cerca anche di integrare alcune esperienze nel campo dell'interazione suono-immagine [5]. Il brano non vuole essere, comunque, né un catalogo sonoro né tantomeno una *summa* esaustiva di queste ricerche: raccoglie soltanto alcune idee e suggestioni nate durante il lavoro sull'analisi.

Sei studi per nastro magnetico

Chromatism è costituito da sei studi distinti per nastro magnetico di circa due minuti ciascuno: inizialmente tali studi dovevano servire a "rappresentare" musicalmente sei dei dodici colori fondamentali del disco cromatico, in modo tale da ricreare, a livello sonoro, alcune suggestioni che all'autore erano state suggerite dal rapporto fra suoni e colori.

Durante la sua lavorazione il brano ha parzialmente perduto questi connotati in favore della creazione di semplici strutture narrative sviluppate all'interno di ogni elemento che lo costituisce e collegate fra loro da un percorso musicale unitario che attraversa l'intera struttura del lavoro. Esiste infatti una sorta di "continuum cromatico" tra i sei pezzi costituenti l'intero brano che, anche se dotati di propria autonomia, dovrebbero essere ascoltati consecutivamente. Si è cercato infatti di far emergere, a livello percettivo, la presenza di una unità formale complessiva, che comunque ci si è preoccupati di riverberare anche all'interno di ogni singolo frammento sonoro.

Ognuno dei sei studi affronta particolari aspetti dell'universo sonoro, come la relazione tra suono e rumore (il primo e l'ultimo), alcune strutture ritmiche o aritmiche (il quarto e il quinto e in parte il sesto) ottenute attraverso l'uso di parametri casuali, la "spazialità" armonica di certe tessiture timbriche (il secondo e il terzo), il fenomeno dei battimenti (ancora il quarto) e così via. Sempre nel segno di una certa continuità del tessuto sonoro, molti elementi musicali (timbrici prima di tutto ma anche ritmici e frequenziali) si ripetono, si fondono e si amplificano da uno studio all'altro.

Da un punto di vista del materiale sonoro di base, all'interno degli studi sono presenti sia suoni elettronici che strumenti acustici campionati. Per quanto riguarda i primi, l'interesse si è incentrato sull'aspetto timbrico con la ricerca di oggetti spesso composti e caratterizzati principalmente da massa tonica e/o complessa [3][6]. Gli strumenti acustici campionati sono stati usati per la creazione di eventi sonori composti, formati dall'assemblamento ritmico di più oggetti semplici, o di "insiemi" sonori con funzione ritmica.

Chromatism è stato composto nel 1992 nello studio della Divisione Musicologica del CNUCE/C.N.R. presso il Conservatorio di Musica di Firenze. Per l'organizzazione formale, la generazione e la gestione degli eventi sonori è stato impiegato il software di composizione automatica Teletau mentre la programmazione timbrica è stata effettuata su due sintetizzatori Yamaha TX81Z e un campionatore Roland S550.

Il brano elettroacustico ha avuto la sua prima esecuzione italiana nel Dicembre 1992 durante il Festival G.A.M.O., al Conservatorio di Musica "L. Cherubini" di Firenze. È stato successivamente eseguito durante il 23º Festival Internazionale di Musica Sperimentale *Synthese 93* di Bourges, nel Giugno 1993.

Bibliografia

- [1] M. Chion: "*L'art des sons fixés*", Ed. Metamkine/Nota-Bene/Sono-Concept, 1991.
- [2] F. Giomi, M. Ligabue: "*Modalities of Signification in Contemporary Music: A Proposal for an Analytical System*", Contemporary Music Review, forthcoming.
- [3] P. Schaeffer: "*Traité des objets musicaux*", Ed. du Seuil, 1966.
- [4] F. Giomi, M. Ligabue: "*Un approccio estetico-cognitivo alla descrizione dell'object sonore*", Atti del II Convegno Europeo di Analisi Musicale (R. Dalmonte, M. Baroni eds.), Università di Trento, 1992.
- [5] M. Aitiani, F. Giomi: "*The artwork Nave di Luce: A Journey into Telematics, Art and Music*", Leonardo, Vol.24, N.2, 1991.
- [6] M. Chion: "*Guide des objets sonores*", INA-GRM/Buchet-Chastel, 1983.

WERVELWIND

David Keane

School of Music
Queen's University
Kingston, Canada K7L 3N6
FAX: 1-416-265-6823
E-mail: keane@qucdn.queensu.ca

Introduction.

Wervelwind (1991) is a work for solo trombone and prepared tape in which the trombonist performs while continuously turning. The turning creates both visual and aural sensations of whirling while it creates an effective dramatic tension through the focused energy of the whirling gesture. Wervelwind is the Flemish word for "whirlwind."

Origin of the work.

Leo Verheyen commissioned Wervelwind with the assistance of a Canada Council Commissioning Grant. Verheyen, one of the best known former students of Vinko Globokar, is himself a well-known proponent of experimental music and is responsible for contemporary music at the Antwerp (Belgium) Conservatory.

Leo requested a work for tape and trombone with a substantial amount of improvisation for the soloist. Consequently, I created an accompaniment generated through emulation of the rich variety of timbres that Leo could

produce on his instrument. The structure of the piece is based on a slow evolution from quiet, air sounds through inharmonic timbres to the more conventional harmonic timbres. The timbre is specified in the score and the player is asked to interact improvisationally with the gestures that are on the tape. At certain points, however, the solo part is fully notated.

Spatialization.

The conspicuously distinguishing feature of Wervelwind is the incorporation of spatial location as a major component of the solo instrument's portion of the performance. Throughout the work, the trombonist rotates so that the bell of the instrument radiates out over 360 degrees while the player is the axis of the rotation.

The use of spatialization as an important parameter in composition has been a major interest of mine for several decades. In addition to a number of works that make use of antiphonal forces, I have written quite a few works locating performers in a circle around the audience [for exam-

ple, Round Dance (1974) for wind ensemble, Tondo (1976) and Orbis (1981) for string orchestra, Corona (1978) for SATB chorus and orchestra, and Henge (1978) for trombone choir]; called for specialized treatments in solo works [for example, Hornbeam (1979) for horn equipped with tubing to direct sound to various locations and Carmina Tenebrarum (1983) an opera for soprano in which the performer rotates in a manner similar to that of the trombonist in Wervelwind]; and have exploited space in the larger "soundscape" of the out-of-doors [for example, Sea-Nauta (1984) for 8 ships in St. John's, Nfld. harbour and Sound Lodge, a computer-based installation for parks].

In Wervelwind, I conceived the rotation as a dominant feature of the musical structure. Leo and I considered motorizing the rotation, but we deemed it safer and more practical for the performer to rotate on his/her feet. This would both offer greater ease of control in altering rotation speed and would provide a more secure basis for the performer to maintain his/her balance--a critical consideration in this case.

Given that it is not practical to read music on a music stand when rotating, the improvisational aspect of the work proved to be especially advantageous. However, the performer makes use of a condensed summary of the instructions for the work on a

small card placed on a lyre. The summary contains an outline of the sequence of events along with timing and tape-cue information. The summary card not only provides an aid to memory, but offers a constant place for the performer to focus his/her eyes as an aid to avoiding dizziness and disorientation while turning.

The sonic result of the rotation is what might be thought of as a slow Doppler effect, although the slow velocity and small distance covered create very little, if any, actual pitch shift. However, because the bell moves laterally and at the same time alternately aims away from and directly at the audience, there are marked cycling amplitude and timbre shifts that strongly reinforce the rotation in the auditory domain to create an effect like an human "Leslie speaker".

It is worth mentioning here, also, that the rotation of the trombone provides an effective visual reinforcement for the auditory cycle. Because little light is necessary for the performer to see music, the performance can be lit with a single spot light placed immediately over the centre point of the rotation. When the performer stands relatively close to the back wall of the stage, the reflections of the light off the rotating trombone offer flowing reflections dancing around the player as an effective visual counterpart to the music.

The Accompaniment.

The tape portion of Wervelwind was assembled using C-Lab Notator 3.1 software running on a Atari Mega ST. All sounds were generated using frequency modulation synthesis on Yamaha tone modules (1-TX802, 1-TX81Z and 3-TX7). These instruments were combined and processed using a Yamaha DMP7 digital mixer, and on-board signal processing supplemented by a Roland DEP5.

Plausible trombone timbres were generated through (1) careful tailoring of voices, (2) layering of a variety of trombone patches and carefully varying the velocity of each, (3) elaborate control of tuning, and (4) use of a great variety of multiple-voice groups.

Integration of solo and accompaniment.

To avoid overwhelming the live performer, I avoided creating rotation effects in the accompaniment, although I make extensive use of the two speaker locations as distinct polarities. Ideally, the speakers are placed just safely beyond the circumference of the trombonist's rotation and about a metre further from the audience than the lateral diameter of the rotation. This effectively melds the solo and tape sounds and, at the same time, offers the performer excellent conditions for monitoring the ac-

companiment and making judgements about balance with the accompaniment.

Therefore, there are effectively four important spatial foci: the LEFT and RIGHT speakers (points C and D in Figure 1), the most FORWARD position of the trombone (point A), and the most rearward position (point B). Position B creates the darkest, most subdued timbre, while position A has the greatest presence. The faster the performer's rotation, the more marked is this timbral contrast. Rotations vary (at the discretion of the performer) from about 20 rotations-per-second to a single rotation-per-second.

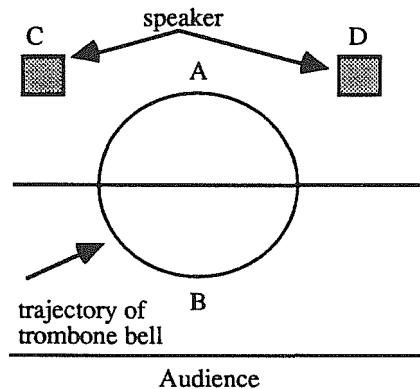


Figure 1 Performance Configuration.

NOTE: a recorded performance of WERVELWIND by Leo Verheyen is available on the compact disc, DAVID KEANE: DIALOGICS [QUM 9301], from the Canadian Music Centre Distribution Service, 20 St. Joseph St., Toronto, Canada, M4Y 1J9].

A COMPOSITION FOR CLARINET AND REAL-TIME SIGNAL PROCESSING: USING MAX ON THE IRCAM SIGNAL PROCESSING WORKSTATION

Cort Lippe

IRCAM, 31 rue St-Merri, Paris, 75004, France
email: lippe@ircam.fr

Introduction.

The composition *Music for Clarinet and ISPW*, by the author, was created using the IRCAM Signal Processing Workstation (ISPW) and the software Max. The piece was commissioned by the Center for Computer Music & Music Technology, Kunitachi College of Music, Tokyo and realized at IRCAM during 1991, and at the Kunitachi College of Music during a composer-in-residency, 1991-92.

From 1988-1991, IRCAM developed a real-time digital processing system, the IRCAM Signal Processing Workstation (ISPW) [1]. Miller Puckette has developed a version of Max for the ISPW that includes signal processing objects, in addition to many of the standard objects found in the Macintosh version of Max [2] [3]. Currently, there are over 40 signal processing objects in Max. Objects exist for most standard signal processing tasks, including: filtering, sampling, pitch tracking, threshold detection, direct-to-disk, delay lines, FFTs, etc. With the ISPW version

of Max, the flexibility with which one creates control patches in the original Macintosh version of Max is carried over into the domain of signal processing.

Prototyping Environment.

The ability to test and develop ideas interactively plays an important role in musical applications. Because of its single architecture, the ISPW is a powerful prototyping and production environment for musical composition [4]. Prototyping in a computer music environment often combines musical elements which traditionally have fallen into the categories of "orchestra" (sound generators) or "score" (control of sound generators). Mainly due to computational limitations, real-time computer music environments have traditionally placed hardware boundaries between "orchestra" and "score": the sound generation is done on one machine while the control is done remotely from another. When developing a synthesis algorithm which makes extensive use of real-time control, it is extremely

helpful, if not essential, to develop the synthesis algorithm and the control software together. This is greatly facilitated when sound generation and control run on the same machine and in the same environment.

Control and Signal Processing.

Real-time signal analysis of instruments for the extraction of musical parameters gives composers useful information about what an instrumentalist is doing. One of the signal processing objects found in Max offers rapid and accurate pitch-detection. In *Music for Clarinet and ISPW*, the incoming clarinet signal is converted via an analog-to-digital converter and analyzed by this pitch-detection algorithm. The pitch tracker outputs MIDI-style pitches which are sent to a score follower [5] (using the *explode* object [6]). As the score follower advances, it triggers the "electronic score" which is stored in event lists. The event lists directly control the signal processing modules. In parallel, compositional algorithms also control the signal processing. These compositional algorithms are themselves controlled by the information extracted from the clarinet input. Thus, the raw clarinet signal, its envelope, continuous pitch information from the pitch detector, the direct output of the score follower, and the electronic score all contribute to control of the compositional algorithms em-

ployed in the piece (see figure below).

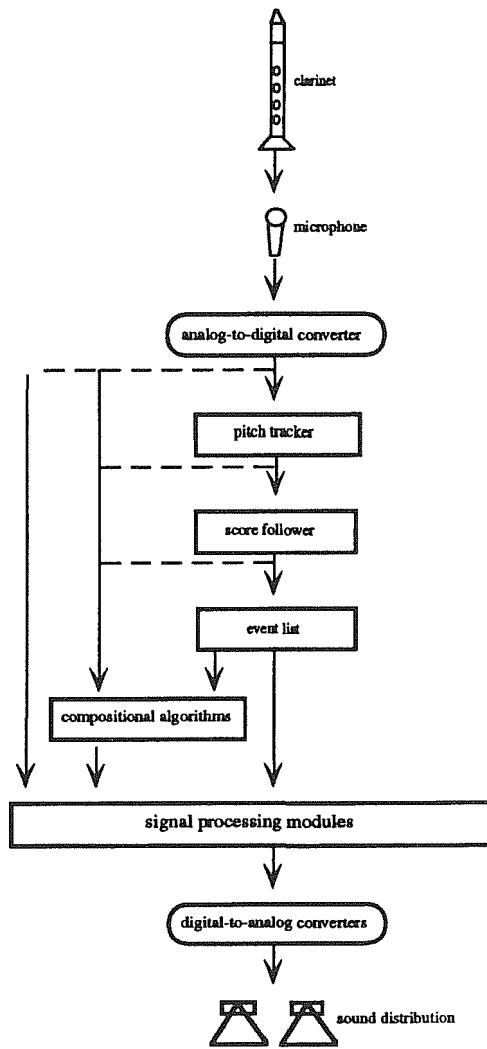


Figure 1. Control and signal processing flow.

The signal processing used in *Music for Clarinet and ISPW* include several standard signal processing modules: reverb, delay, harmonizing, flanging, frequency

shifting, spatializing, and frequency/amplitude modulation. Several non-standard sampling techniques are used also, including a time-stretching algorithm, developed by Puckette, which allows for the separation of sample transposition and sample duration. Thus, one can slow down a sample playback while maintaining the original pitch, or change the pitch of a sample playback without changing its duration. Another sampling technique, a kind of "granular" sampling developed from techniques described by Xenakis [7] and Roads [8] for sound synthesis, is also used. Ten-second sound samples can be played back in a variety of ways and orderings, taking approximately 20-millisecond "sound grains" of the sample at a time. (All of the samples are made up of clarinet phrases sampled in real-time during the performance of the piece.) Finally, using an automated signal crossbar (similar to a studio patch-bay) to connect modules to each other, signals can be sent from the output of practically every module to the input of every other module. This signal crossbar maximizes the number of possible signal paths and allows for greater flexibility when using a limited number of signal processing modules [9] (see figure below).

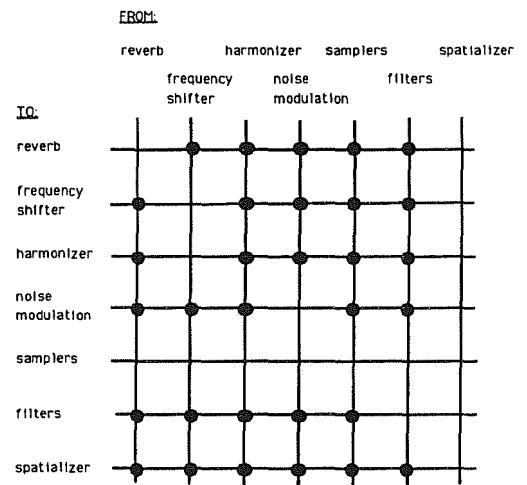


Figure 2. Crossbar of interconnections among signal processing modules.

Real-time Continuous Control Parameters

Real-time audio signal analysis of acoustic instruments, for the extraction of continuous control signals that carry musically expressive information, can be used to drive signal processing and sound generating modules, and can ultimately provide an instrumentalist with a high degree of expressive control over an electronic score [10]. In the frequency domain, pitch tracking can be used to determine the stability of pitch on a continuous basis for recognition of *pitch-bend*, *portamento*, *glissando*, *trill*, *tremolo*, etc. In the amplitude domain, envelope following of the continuous dynamic envelope for articulation

detection enables one to determine *flutter-tongue*, *staccato*, *legato*, *sforzando*, *crescendo*, etc. In the spectral domain, FFTs, pitch tracking, and filtering can be used to track continuous changes in the spectral content of sounds for detection of *multiphonics*, *inharmonic/harmonic ratios*, *timbral brightness*, etc. High-level event detection combining the analyses of frequency, amplitude, and spectral domains can provide rich control signals that reflect subtle changes found in the input signal.

The Musician's Role.

The dynamic relationship between performer and musical material, as expressed in the musical interpretation, can become an important aspect of the man/machine interface for the composer and performer, as well as for the listener, in an environment where musical expression is used to control an electronic score. The richness of compositional information useful to the composer is obvious in this domain, but other important aspects exist: compositions can be fine-tuned to individual performing characteristics of different musicians, intimacy between performer and machine can become a factor, and performers can readily sense consequences of their performance and their musical interpretation.

References.

- [1] E. Lindemann, M. Starkier, and F. Dechelle. "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features." In S. Arnold and G. Hair, eds. *Proceedings of the 1990 International Computer Music Conference*. San Francisco: International Computer Music Association, 1990.
- [2] M. Puckette. "The Patcher." In C. Lischka and J. Fritsch, eds. *Proceedings of the 1988 International Computer Music Conference*. San Francisco: International Computer Music Association, 1988.
- [3] M. Puckette. "Combining Event and Signal Processing in the Max Graphical Programming Environment." *Computer Music Journal* 15(3):68 - 77, 1991.
- [4] C. Lippe *et al*, "The IR-CAM Musical Workstation: A Prototyping and Production Tool for Real-Time Computer Music." *Proceedings, 9th Italian Colloquium of Computer Music*, 1991, Genoa.
- [5] M. Puckette, "EXPLODE: A User Interface for Sequencing and Score Following." In S. Arnold and G. Hair, eds. *Proceedings of the 1990 International Computer Music Conference*. San Francisco: International Computer Music Association, 1990.
- [6] M. Puckette and C. Lippe. "Score Following in Practice." In *Proceedings of the 1992 International Computer Music Conference*.

ence. San Francisco: International Computer Music Association, 1992.

[7] I. Xenakis. *Formalized Music*. Bloomington: Indiana University Press. (Pendragon, 1991) 1971.

[8] C. Roads. "Automated Granular Synthesis of Sound." *Computer Music Journal* 2(2):61 - 62, 1978.

[9] C. Lippe and M. Puckette. "Musical Performance Using the IRCAM Workstation." In B. Alphonse and B. Pennycook, eds. *Proceedings of the 1991 International Computer Music Conference*. San Francisco: International Computer Music Association, 1991.

[10] D. Wessel, D. Bristow and Z. Settel. "Control of Phrasing and Articulation in Synthesis." *Proceedings of the 1987 International Computer Music Conference*. San Francisco: International Computer Music Association, 1987.

IHADA PER SASSOFONO TENORE E SUONI SINTETICI

Matteo Pennese

AGON <Acustica Informatica Musica>
Piazzale Egeo, 5 - 20126 Milano
Tel. 02-64429289
Fax 02-64422724

Abstract

Through its numerical treatment, the composer can create the sound.

But can we consider the timbre, using a loved-serialist term, a *parameter*?

In other words, has timbre the capacity to take part in the musical form and, if so, in which degree?

Starting from McAdams's classification regarding the attributes of the form-carriers elements, *ihada* represents an effort to understand which is the timbre limit, taken as a composite element. The aim has been to create a timbre-model, considered as the "heart" of the material, and later on creating tension or distension states through its distortion.

Premessa

Può il timbro, utilizzando un termine caro alla musica seriale, considerarsi un parametro?

In altre parole, ha il timbro la capacità di contribuire alla forma musicale, intendendo questa come ciò che si percepisce

e che non necessariamente come la struttura immaginata dal compositore?

E se la risposta risulterà affermativa, in quale grado?

Stephen McAdams [1] propone l' identificazione dei c.d. "elementi portatori di forma" in base a sei criteri, qui di seguito riassunti:

1. differenziazione in categorie percettive discrete;
2. organizzazione delle categorie percettive in modo tale da far sì che le loro relazioni siano d'ordine funzionale;
3. capacità delle differenze di natura e forza fra le relazioni funzionali, di creare stati di tensione e distensione;
4. qualità distintive d' una categoria, qualità di relazioni fra le categorie ovvero modo in cui queste ultime si collegano con le altre, quali attrattivi dell' attenzione;
5. disponibilità delle categorie, delle relazioni funzionali e dello ordinamento di un sistema di classificazione, di essere apprese da un ascoltatore;
6. mantenimento di un certo grado di invarianza delle rela-

zioni fra categorie attraverso le differenti classi di trasformazione.

Ma se ci rivolgiamo al timbro, già al primo punto della sopra esposta classificazione incontriamo grossi ostacoli.

Se altezza e timbro sono attributi di un medesimo segnale acustico, tuttavia vi sono fra loro sostanziali differenze.

Da un lato il "continuum" delle altezze ci porta ad una categorizzazione discreta; dall'altro lato altrettanto non può dirsi del timbro. Ulteriore fattore che impedisce la discretizzazione del timbro è costituito dalla sua multidimensionalità, ovvero l' impossibilità di definire il timbro sulla base di categorie riconoscibili con precisione nei più svariati contesti. Ciò comporta l' assenza della relazione di ottava, fattore che facilita l' apprendimento e la memorizzazione.

Insomma, se altezze, intensità e durate sono misurabili rispettivamente in Hertz, millisecondi e livello d' ampiezza, il timbro lo si può, tutt' al più, identificare nel complesso delle tre componenti citate. Il timbro non è un componente ma un composto.

E la molteplicità degli elementi complica considerevolmente il trattamento dei fenomeni percettivi (Miller 1952 [2]).

Vi è, tuttavia, una chiave di volta. L' universo timbrico non può, nonostante tutto, definirsi come totalmente continuo. Esistono delle *differenze*.

Prima dei modelli, delle gerarchie, delle strutture, delle forme - in altre parole prima della cultura - esistono delle differenze. Vi sono realtà acustiche ineludibili: i suoni con spettro armonico sono più distensivi di quelli a spettro inarmonico, così come un' ottava sarà sempre diversa da una terza minore e, ancora, un' ottava non accordata produce il fenomeno dei battimenti, venendo percepito come intervallo più "rugoso" rispetto ad un' ottava accordata (i battimenti e la rugosità sono dovuti ad un' interferenza, nell' orecchio interno, d' una medesima banda critica, Pomp [3]).

In campo strumentale, un suono di violino risulta più "brillante" di quello di una viola suonata alla medesima altezza ed intensità (la "brillantezza" riproduce, a livello di percezione, la ripartizione dell' energia sullo spettro, Wessel [4]).

Possiamo individuare, quindi, due livelli: nel primo parliamo di grado di tensione, nel secondo, invece, prendiamo in considerazione la dissonanza e la consonanza, e da qui inizia la cultura.

ihada : obbiettivo e forma

In base a queste premesse, dunque, il mio primo obbiettivo compositivo dei suoni di sintesi, è stato quello di poter lavorare con uno strumento in grado di creare un nucleo timbrico fisso dal quale, tramite la modifica di

determinati parametri ci si potesse allontanare o riavvicinare, senza perdere contatto con il "cuore" timbrico.

In *ihada* si può dunque individuare un timbro-modello, una sorta di prototipo, udibile come un suono di campana.

Distinguiamo cinque sezioni:

I sezione: reiterazione di un accordo, la cui struttura rimane identica venendo modificate, per contro, le zone di risonanza, che vanno a rivelare una melodia nascosta fra le pieghe timbriche. Progressivamente l' accordo acquista, da un lato, un transitorio d' attacco sempre più breve, avvicinandosi al timbro-modello.

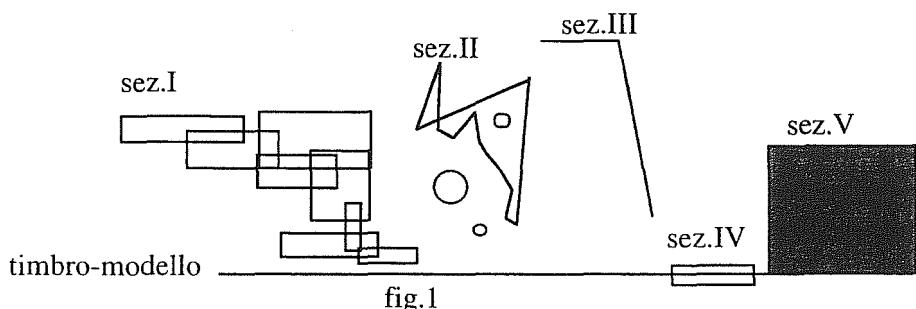
II sezione: compaiono delle linee, dei "grumi sonori", come se gli accordi iniziali si fossero sia contratti che disintegriti.

III sezione: cresce la tensione timbrica. I suoni divengono to-

talmente inarmonici esplodendo, alla fine, in un suono percussivo in fortissimo, che a sua volta si disintegra. Sulla coda di questo suono, in pianissimo, compare un dolce rintocco di campana, che segna l' inizio della sezione seguente.

IV sezione: si presenta il timbro-modello. La sezione, inoltre, è segnata da una pulsazione regolare, che contribuisce alla massima distensione.

V sezione: dagli accordi della sezione precedente si espande una fascia lontana, che richiama il fantasma di uno scampanio, su cui ricama il suo canto lo strumento solista. Per un attimo ricompaiano forme variate del timbro-modello, risucchiate dalla fascia che lentamente si estingue. Questa struttura si può raffigurare graficamente nella figura 1,



da cui si evince chiaramente la maggiore o minore distorsione rispetto al timbro-modello delle varie sezioni timbriche.

Conclusione

Credo che il timbro, ora come ora, rivesta una posizione piuttosto ibrida nel vocabolario del compositore. Se da un lato è una categoria percettiva, come l' altezza o la durata, d' altro lato ne

viene impedita la sua categorizzazione concettuale.

Appare come elemento di indubbio fascino per il compositore, poichè rappresenta la superficie, il colore, la "pelle" del suono.

E, d'altra parte, si dimostra strumento incapace di reggere da solo un cammino formale convincente, imponendo una regressione verso il puro materiale sonoro nella sua varietà fenomenologica, accatastato in successioni spesso banali.

Il trattamento numerico del suono, dunque, apre immensi spazi al compositore di oggi; tuttavia non si garantisce la qualità musicale in virtù di un materiale acustico sempre più variegato e ricco.

ture", Computer Music Journal, vol.3, n° 2.

Riferimenti bibliografici

[1] S. McAdams, aa.vv.: "*Le timbre, métaphore pour la composition*", I.R.C.A.M., Christian Bourgois éditeur, pp. 164-165-166-167, 1991.

[2] G.A. Miller, "*The magical number seven, plus or minus two: some limits on our capacity for processing information*", Psychological Review, n° 63, pp. 81-97.

[3] R. Plomp, W.J.M. Levelt, "*Tonal consonance and critical bandwidth*", Journal of the Acoustical Society of America, n° 46, 1965.

[4] D.L. Wessel, "*Timbre space as a musical control struc-*

