Analisi e sintesi in tempo reale mediante riconoscimento timbrico

Marco Matteo Markidis
nonoLab, Parma
mm.markidis@gmail.com

José Miguel Fernández
IRCAM, Parigi
jose.miguel.fernandez@ircam.fr

ABSTRACT

Questo articolo descrive *path*∼, un external per *Pure Data* [1] che implementa un sistema di analisi e sintesi concatenativa basata su un *corpus* audio, ossia una collezione di file audio. Il suono è acquisito ed analizzato mediante un insieme di descrittori audio estratti in tempo differito. Il processo di sintesi in tempo reale avviene mediante riconoscimento di similarità tra i grani audio dei *corpora sonora* analizzati e il segnale in tempo reale.

La catena di processo avviene, in tempo differito, segmentando il file e generando i grani, eseguendo successivamente un'analisi di ognuno di questi ed estraendone i descrittori che contribuiscono alla creazione di un albero *k*-dimensionale e una lista di *k*-primi vicini per ogni grano generato, dove ogni grano è rappresentato da un insieme di punti. La sorgente entra in *path*∼, ne vengono estratti i descrittori in tempo reale ed il grano più simile viene trovato nell'albero. Partendo dall'elemento trovato ed usando la sua lista di primi vicini si rende disponibile un insieme di grani per la sintesi.

1. INTRODUZIONE

Attraverso le tecniche di sintesi del suono si possono produrre innumerevoli timbri. Alcuni di questi non prendono ispirazione da suoni esistenti; altri, attraverso una fase di analisi, possono portare alla ricostruzione timbrica di suoni esistenti.

Storicamente la sintesi granulare compare tra le prime applicazioni di produzione e ricostruzione sonora [1] . A differenza di altre tecniche di sintesi, ormai cristallizzate nel tempo, la sintesi granulare continua ad avere un percorso di sviluppo lasciando aperte discussioni su varie tecniche, in particolare la sintesi concatenativa ed i mosaici audio. La differenza principale tra queste tecniche e la sintesi granulare è che nella sintesi concatenativa i grani sono legati fra loro solitamente da un'analisi precedentemente fatta su un file audio. Questa relazione può essere basata su differenti metodi ed è solitamente guidata dall'analisi.

L'idea base della sintesi mediante riconoscimento timbrico è di estrarre descrittori da un file audio, di costruire uno spazio timbrico e di introdurre una metrica affinché il concetto di distanza diventi significativo. Tramite l'estrazione di descrittori, le similarità tra i suoni sono sottolineate da una vicinanza geometrica nello spazio dei descrittori. Questo spazio è multidimensionale e la sua dimensionalità dipende dal numero e dal tipo di descrittori usati nell'analisi. Per eseguire un corretto riconoscimento, diverse operazioni vengono eseguite in tempo differito: in particolare, la costruzione del *database* multidimensionale fatto con i dati estratti dall'analisi dei descrittori. In tempo reale, un suono entrante è analizzato mediante lo stesso insieme di descrittori usato per l'analisi in tempo differito e la ricerca del primo vicino è condotta sul database. La fase di sintesi si riferisce alla parte finale dell'algoritmo, in cui partendo dall'elemento più simile un insieme di grani fatto dai k elementi primi vicini è disponibile per essere eseguito in un processo di granulazione.

La motivazione principale di questo lavoro è sviluppare un sistema di analisi e sintesi concatenativa. L'idea è stata di sviluppare questo sistema in ambiente *Pure Data*, in maniera da poter dare al compositore o al regista informatico la possibilità di integrare facilmente questo sistema nei propri programmi dedicati al processamento del suono in tempo reale. Lo scopo del progetto inoltre è stato quello di creare uno strumento *open-source* specifico per la musica contemporanea dato che in altri ambienti di lavoro commerciali progetti e lavori simili erano già presenti [2] .

Il contributo principale del lavoro è di provvedere alle funzionalità richieste dalla sintesi concatenativa in un singolo oggetto, senza l'utilizzo di librerie ausiliari per l'analisi e la sintesi. In questo modo, *path*∼ rappresenta un sistema compatto che incorpora internamente tutti i passaggi distinti necessari per la sintesi concatenativa. Inoltre, il design interno è pensato per un suo utilizzo in tempo reale. Oltre all'analisi, altre operazioni sono eseguite in tempo differito, come l'ordinamento del database o il calcolo dei primi vicini, necessario per la parte di sintesi dell'algoritmo. La ricerca su un database grande, ossia su file audio lunghi, è eseguita in un tempo ragionevole; una discussione più dettagliata è esposta in seguito.

L'articolo è organizzato come segue; una sezione descrive i lavori collegati, focalizzando l'attenzione sui progetti presenti in Max/MSP e Pd. La successiva descrive i metodi utilizzati in *path*∼, con enfasi particolare sull'algoritmo proposto e alla sua implementazione. I risultati, sia artistici che tecnici, vengono presentati nella quarta sezione. Infine verranno discussi i futuri lavori e ricerche.

[1] *Concret PH* di I.Xenakis ne è una delle prime applicazioni musicali.

[2] Una simile limitazione è sottolineata anche da William Brent in [2].

2. LAVORI COLLEGATI

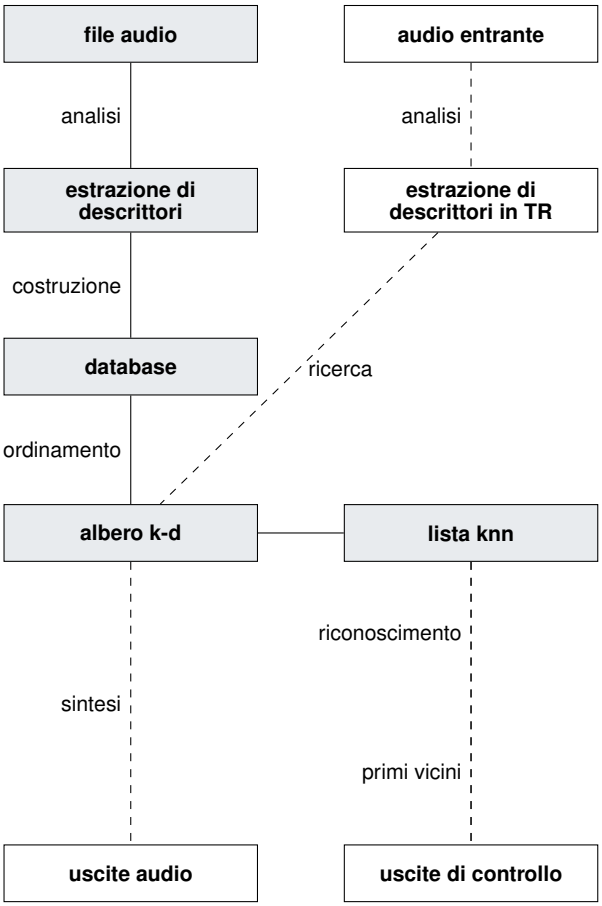
Dopo i lavori pioneristici di Iannis Xenakis, l'attenzione dei compositori e dei ricercatori sulla sintesi granulare e sulla granulazione di suoni campionati è cresciuta al punto che, attualmente, la sintesi granulare è una delle tecniche più usate nella composizione di musica elettronica.

Diversi progetti sono stati sviluppati usando tecniche che involvono sintesi concatenativa a partire dagli anni Duemila. In particolare, la sintesi concatenativa è stata esplorata con *CataRT* [3], un insieme di patch per Max/MSP che utilizza diverse estensioni, come *FTM*, *Gabor* e *MnM*. L'idea è quella di caricare descrittori audio da file SDIF [3], o di calcolare descrittori sul momento, e di creare uno spazio dei descrittori che usa grani corti come punti. L'utente può scegliere due descrittori e usando un'interfaccia grafica implementata *CataRT* disegna una proiezione bidimensionale dei grani nello spazio dei descrittori, dove l'utente può navigare ed esplorare lo spazio dei descrittori tramite la posizione del mouse. Attraverso l'interfaccia grafica, la natura espressiva dei grani può essere sottolineata. L'architettura di *CataRT* può essere divisa in cinque parti: analisi, dati, selezione, sintesi ed interfaccia. A parte le utilità grafiche, il sistema utilizza altri strumenti *Max/MSP* come la libreria *Gabor* per la sintesi; inoltre le strutture dati *FTM* vengono utilizzate per la gestione dei dati. Un brano per banjo ed elettronica di Sam Britton è stato composto con *CataRT*.

Un altro progetto *Max/MSP* è stato la realizzazione di Caméléon Kaléidoscope di Marco Antonio Suárez Cifuentes [4] per la *Biennale Musiques en Scéne* di Lione nel marzo 2010. Per questa produzione, una combinazione di *FTM*, *Gabor*, *MnM* e librerie *MuBu* è stata implementata. *MuBu* è un contenitore multitraccia che rappresenta uno stream di dati omogenei allineati temporalmente simile allo stream di dati rappresentato dallo standard *SDIF* [5].

Nel mondo *Pd*, alcuni lavori sono stati proposti sul tema dell'analisi timbrica e sulla classificazione sonora. A parte l'oggetto *bonk~* [6], che è un oggetto per la detenzione delle soglie audio, uno dei progetti più importanti è la libreria *timbreID* [2] sviluppata da William Brent. Questa libreria è una collezione di external per l'estrazione di descrittori in tempo reale e in modalità batch. In particolare, oltre alla presenza di descrittori a basso livello, ossia descrittori a singolo valore, alcuni descrittori ad alto livello, ossia una lista di valori, sono stati implementati, come il *mel frequency cepstrum* e il *bark frequency cepstrum*. L'autore mostra che l'utilizzo di descrittori ad alto livello ha un riconoscimento migliore rispetto a quelli a basso livello [7]. Inoltre, mostra anche come l'utilizzo di una combinazione di descrittori, in particolare un descrittore ad alto livello unito a descrittori a basso livello, aumenta la percentuale di riconoscimento a tempo di analisi fissato rispetto all'utilizzo di descrittori singoli. Questa peculiarità ha un grande interesse nella sintesi basata sulla similarità sonora, grazie alla possibilità di un miglior riconoscimento tra il suono

[3] Sound Description Interchange Format.



path~:
gli oggetti bianchi e collegati da linee tratteggiate si riferiscono alle operazioni in tempo reale,
gli oggetti grigi e collegati da linee continue si riferiscono a processi in tempo differito.

di input e gli elementi presenti nel database. In aggiunta ai descrittori, un oggetto chiamato *timbreID*, il quale permette una gestione delle informazioni del database, è introdotto.

Infine è stato presentato un recente lavoro audio-video, che implementa un'interfaccia a controllo gestuale [8]. Questo progetto è basato su Pd, sulla libreria per computer grafica *GEM* e sulla libreria *timbreID*.

3. METODI

path~ esegue prima l'analisi in tempo differito. Dopo aver caricato i campioni, o in formato di file audio mono dal disco rigido o da un array di Pd, *path~* segmenta i corpora audio creando l'insieme di grani. Due tipi di segmentazione sono possibili; il primo è basato su una finestra, ossia data una finestra *path~* taglia i corpora audio ogni data finestra ed esegue l'analisi; tutti i grani avranno quindi la stessa lunghezza. La seconda frammentazione possibile è basata su una soglia, ossia viene eseguita una detenzione degli eventi sui corpora audio; questa frammentazione rispetta la natura gestuale del contenuto del file audio. In questo caso, i grani avranno lunghezze diverse. Successi-

vamente vengono estratti i descrittori da tutti i grani ed i dati raccolti formeranno un database. Abbiamo scelto due strategie per i descrittori.

L'implementazione di default è costituita da un descrittore ad alto livello più un insieme di due descrittori di basso livello, mentre la seconda esclude il descrittore ad alto livello lasciando gli altri. Il descrittore ad alto livello è un mel frequency cepstrum; i descrittori a basso livello sono un centroide spettrale e un descrittore di ampiezza. La computazione dei Coefficienti Mel Frequency Cepstral (MFCC) richiede un banco di filtri passabanda che si sovrappongono equispaziati sulla scala di mel. Un mel è l'unità della scala definita come:

$$Mel(f) = 2595 \times log_{10}(1 + \frac{f}{700}), \qquad (1)$$

dove f è la frequenza in Hz. Infine, per ottenere i MFCC utilizziamo una trasformata discreta del coseno:

$$MFCC_i = \sum_{k=0}^{N-1} X_k cos[i(k + \frac{1}{2})\frac{\pi}{N}], \qquad (2)$$

dove $i = 0, 1, \ldots, M-1$. M è il numero dei coefficienti cepstrali, N il numero dei filtri e X_k è la potenza logaritmica in uscita del k-esimo filtro. L'utente può scegliere la frequenza spaziale di *mel* durante la creazione dell'oggetto. Variando questo paramento, il numero dei filtri risultanti cambierà. Questo significa che la dimensione del nostro spazio timbrico cambierà in accordo a questo parametro. Il valore di default della spaziatura di *mel* è 250 *Hz*, che risulta in uno spazio timbrico a 14 dimensioni, che è il numero dei *MFCC*, più 2, che è la dimensionalità risultante del centroide spettrale e del descrittore di ampiezza: ogni grano verrà quindi rappresentato da un insieme di 16 numeri, che saranno le coordinate del grano nello spazio timbrico.

Il centroide spettrale è definito come il centro di massa della magnitudine spettrale:

$$SC = \frac{\sum_{k=0}^{N/2} f(k)|X(k)|}{\sum_{k=0}^{N/2} |X(k)|} \qquad (3)$$

con $f(k)$ la frequenza del k-esimo bin.

Il descrittore di ampiezza è di default un'ampiezza in scarto quadratico medio; può essere cambiato nel picco dell'ampiezza.

L'estrazione dei descrittori è un'operazione che deve essere eseguita su un numero di campioni che corrisponde a una potenza di due. Nel caso della frammentazione basata su finestra, la lunghezza della finestra di analisi è uguale alla finestra di taglio, che è una potenza di due. Nel caso di segmentazione basata sulla soglia, l'estrazione di descrittori avverrà su un numero di campioni arrotondato alla precedente potenza di due rispetto alla lunghezza totale del grano. Questa operazione forma un database, che verrà successivamente ordinato in un albero k-dimensionale. Questo ordinamento è utile per l'ottimizzazione della ricerca del primo vicino in tempo reale. Ogni elemento avrà un indice identificativo, che corrisponderà alla posizione del grano nel file audio. Infine *path*$\sim$ costruisce una lista di k primi vicini, che consiste nei grani disponibili per la sintesi. L'algoritmo per la creazione della lista dei primi vicini è un *Quickselsort*.

In tempo reale, *path*$\sim$ esegue una estrazione di descrittori sulla sorgente di input. Cerca nell'albero k-dimensionale l'elemento che minimizza la distanza euclidea e, accedendo alla sua lista di primi vicini, sintetizza un grano e fornisce le informazioni di controllo. Questa operazione è accurata al campione; infatti l'analisi è fatta sulla richiesta dell'utente ed è indipendente dal *blocksize*. Le informazioni di controllo sono l'indice del grano riconosciuto nel database, la lista con gli indici dei *k*-primi vicini desiderati, il numero totale di grani e il numero dei grani che stanno suonando. Oltre al contenuto sonoro, le caratteristiche principali dei grani creati sono l'ampiezza, la lunghezza, l'inviluppo e la concatenazione. Queste caratteristiche possono essere controllate direttamente dall'utente tramite specifiche funzioni. Il termine concatenazione si riferisce alla lunghezza del treno di grani, ossia al numero di grani che verranno suonati a ogni richiesta dell'utente. In questo modo, un treno di grani può partire creando tessiture sonore oltre al grano trovato dal riconoscimento dei dati estratti dai descrittori.

```
preset @ intro ;
threshold = -1.;
window = 512;
concatenate = 1;
amp = 1;
hopsize = 100;
weight = 1;
envelope = 0;
# questo  e   un commento ;
endpre

preset @ sec1 ;
window = 2048 ;
hopsize = 250 500;
concatenate = 2;
# envelope = 2 ;
endpre

preset @ sec2 ;
threshold = 0.25;
concatenate = 3;
amp = 0.2 0.4;
hopsize = 100;
envelope = 2;
# questo  e   un commento ;
endpre

preset @ final ;
window = 2048 ;
hopsize = 250 500;
# envelope = 2 ;
endpre

# token possibili:
# window : int potenza di due
# threshold: float (0,1)
# concatenate: int>0
# amp: float>0, un valore = all, due = random () , tre lista
# hopsize: int>0, un valore = all, due random () , tre lista
# weight: float (0,1) lista
# envelope: int, 0 hann, 1 hamm, 2 blackman, 3 cosine, 4 rectangular (constant envelope)
```

Preset file example.

Inoltre abbiamo implementato un interprete di script pensato per essere un sistema di gestione di preset. In questo modo, l'utente può scrivere uno script con una data sintassi e *path*∼ lo carica e crea una lista di preset che può essere utilizzata con una funzione dedicata. Opzionalmente, l'utente può specificare per ogni preset un nome identificativo. Oltre all'interprete, un semplice debugger è presente. Inoltre la lista completa dei preset insieme ai nomi identificativi dei preset può essere visualizzata in un editor grafico che può essere richiamato direttamente da *path*∼.

path∼ supporta una strategia di multithreading. In questo modo, un thread worker esegue l'analisi in tempo differito mentre il thread principale di Pd continua la sua esecuzione. Con questa strategia, l'analisi in tempo differito può essere eseguita senza interrompere il flusso audio.

4. RISULTATI

4.1 Latenza

Nell'audio in tempo reale, uno dei problemi cardine è la latenza, definita come il ritardo temporale tra un input di arrivo e l'output di risposta. Abbiamo quindi studiato il tempo impiegato da *path*∼ tra l'input dato dall'utente e la risposta dei grani. La nostra stima per il limite di latenza accettabile per il tempo reale è di 6 *ms*. Sulla macchina di uno degli autori, ossia un Mac BookPro Early 2011 13" 2.7 GHz con Pd Vanilla 0.46.6, su un database di grandezza nell'ordine dei 30K ottenuto usando un file audio lungo dieci minuti con una finestra di analisi di 23.22 ms, la ricerca nel database impiega di media meno di 2 ms, mentre l'analisi in tempo differito impiega di media meno di un minuto.

4.2 Applicazioni nella composizione contemporanea

L'utilizzo di sistemi di analisi in tempo reale per il suono è un concetto chiave nell'interazione tra strumenti acustici ed elettronica. Se negli ultimi anni diversi metodi di analisi del segnale, ad esempio l'estrazione di descrittori audio, sono stati implementati, questi metodi non sono stati esplorati in tutte le loro possibilità.

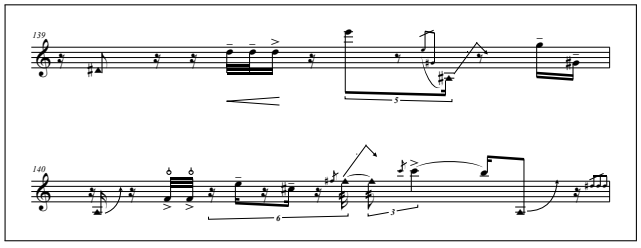


Figure 1: Estratto della partitura di *Dispersion de trajectoires* che abbiamo usato per le nostre applicazioni.

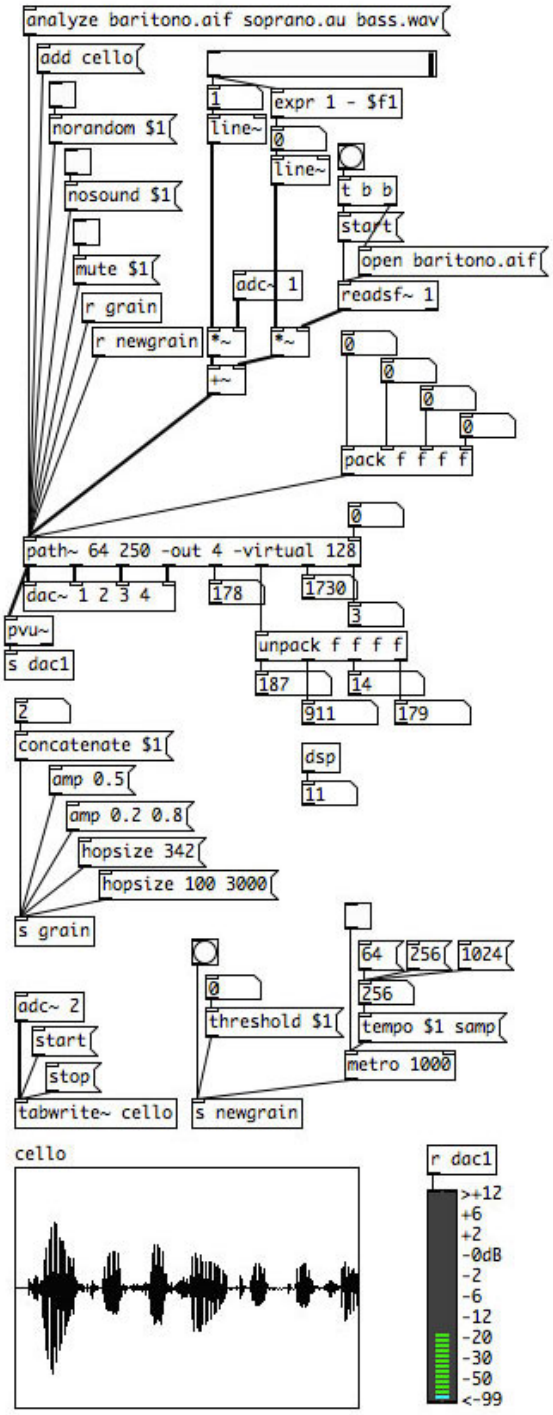


Figure 2: *path*∼ in un patch d'applicazione.

path∼ dà l'opportunità al regista informatico di sperimentare e di creare diversi tipi di materiale elettroacustico in tempo reale o di generare tessiture elettroacustiche. La possibilità di creare ed analizzare diversi file audio creando database come la possibilità di cambiare dinamicamente le caratteristiche dei grani come l'ampiezza, la durata e l'inviluppo attraverso un sistema di preset basato su script dà a *path*∼ una grande versatilità e flessibilità. Inoltre l'implementazione di una funzione asincrona basata su una soglia, che permette di innescare grani quando sono vicini nello spazio timbrico alla sorgente in ingresso unito alla possibi-

lità di una frammentazione basata sugli eventi permette di creare tessiture concatenative estremamente articolate.

Una grande estensione è offerta dal poter analizzare un array di Pd; in questo modo l'analisi può essere eseguita su materiale appena registrato. Questo permette di registrare suoni da utilizzare per l'analisi in *path*∼ direttamente durante le prove o i concerti.

Inoltre, *path*∼ utilizza un sistema di uscite audio vettorializzato. Durante la creazione dell'oggetto, l'utente può specificare il numero di uscite e il numero di canali virtuali, ossia il massimo numero di grani che può essere suonato contemporaneamente. La motivazione principale di questo controllo dinamico è il controllo di configurazioni diverse di diffusori, per esempio durante le sessioni in studio o durante una performance live, e la possibilità di eseguire lo stesso brano musicale su un numero e una configurazione diversa di diffusori.

Infine, *path*∼ può escludere la parte di sintesi dall'algoritmo lasciando accesso solo alle informazioni di controllo. In questo modo, applicazioni che non fanno uso della sintesi concatenativa ma soltanto della parte di analisi possono essere eseguite con risparmio di CPU.

5. DISCUSSIONE E CONCLUSIONI

Lo scopo primario di questo lavoro è la produzione di brani di musica contemporanea per strumenti acustici ed elettronica. Per raggiungere questo scopo, diverse funzionalità sono pianificate. In particolare, una versione polifonica sarebbe utile per un contesto di ensemble e un controllo maggiore sulla spazializzazione e sulla localizzazione dei singoli grani nello spazio sarebbe desiderabile. Inoltre, un miglioramento dell'editor unito a una maggiore configurazione dei preset è in programma. Un crescente interesse sarà dedicato ad un'implementazione di un'analisi a cluster e sul riconoscimento, cercando di sviluppare alcune strategie specifiche per le tecniche strumentali estese e per utilizzare grani con lunghezze diverse.

Nonostante il lavoro sia ancora in divenire, gli autori stanno usando *path*∼ per i loro progetti sonori. In particolare, Marco Matteo Markidis sta usando *path*∼ nel suo duo di improvvisazione radicale Adiabatic Invariants. Attualmente, in *Cattedrali di Sabbia* ne è stato fatto un uso massiccio.

path∼ è stato testato su sistemi OS X e Linux usando Pd Vanilla versione stabile 0.46.6. *path*∼ è un software open source ed è rilasciato sotto licenza GPLv3. È disponibile su Deken, il gestore di externals per Pure Data. L'attuale versione è la 1.5.0 alpha.

6. RINGRAZIAMENTI

Vorremmo ringraziare Stefano Markidis per i suoi consigli e suggerimenti; Martino Traversa e Fondazione Prometeo per il supporto e l'incoraggiamento datoci in questo periodo. Un ringraziamento speciale va infine a Giuseppe Silvi, per il suo aiuto e la sua dedizione, a Daniele Pozzi, senza il quale il nostro interesse in questo lavoro non sarebbe nato, e a Liarss Production, nella persona di Luca Gazzi, che ha dedicato al nostro progetto la sua pazienza e molte ore del suo tempo.

7. BIBLIOGRAFIA

[1] M. Puckette, "Pure data: another integrated computer music environment," in *Proceedings of the International Computer Music Conference (ICMC)*, 1996.

[2] W. Brent, "A timbre analysis and classification toolkit for pure data," in *Proceedings of the International Computer Music Conference (ICMC)*, 2009.

[3] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2006.

[4] N. Schnell, M. A. S. Cifuentes, and J.-P. Lambert, "First steps in relaxed real-time typo-morphological audio analysis/synthesis," in *Proceedings of an International Conference on Sound and Music Computing (SMC)*, 2010.

[5] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "Mubu & friends - assembling tools for content based real-time interactive audio processing in max/msp," in *Proceedings of an International Computer Music Conference (ICMC)*, 2009.

[6] M. Puckette, T. Apel, and D. Zicarelli, "Real-time audio analysis tools for pd and max," in *Proceedings of the International Computer Music Conference (ICMC)*, 1998.

[7] W. Brent, "Cepstral analysis tools for percussive timbre identification," in *Proceedings of International Pd Conference (PdCon09)*, 2009.

[8] J. Gossmann and M. Neupert, "Musical interface to audiovisual corpora of arbitrary instruments," in *Proceedings of New Interfaces for Musical Expression (NIME)*, 2014.