Ambiophonic Reverberation

Research Notes

Paradisi Francesco
Conservatorio S. Cecilia di Roma
francesco.paradisi10 @ gmail.com

Giuseppe Silvi
Conservatorio Nicolini di Bari
silvi.giuseppe @ docenticonsba.it

Edoardo Staffa
Conservatorio S. Cecilia di Roma
edoardo.staffa1 @ gmail.com

Abstract

Il riverbero digitale nei modelli indicati da Schroeder nel testo del 1962 presenta, a quasi sessanta anni di distanza, innumerevoli spunti di sperimentazione e riflessione tanto in ambito tecnico (informatico ed acustico) quanto in ambito musicale (compositivo ed interpretativo). Nelle pratiche di bilanciamento tra suono acustico ed elettroacustico il riverbero gioca un ruolo sostanziale ed il suggerimento conclusivo di Schroeder di utilizzare un riverbero ambiofonico per la sala da concerto non risulta essere compreso e diffuso nella pratica musicale contemporanea.

Generalmente si accomuna la ricerca di Schroeder con sistemi commerciali introdotti negli anni ottanta, mentre la sua ricerca appare essere molto più estesa ed in prospettiva concertistica, quindi elettroacustica con riferimenti alla correzione adattiva degli spazi che risulta essere un ambito di ricerca ancora non completamente soddisfatto.

La ricerca qui presentata è in una fase di primo sviluppo. Particolare attenzione è stata dedicata allo sviluppo di pratiche di analisi dei segnali per testare ogni singola fase di sviluppo degli algoritmi proposti da Schroeder e perfettamente documentati.

SCHROEDER - NATURAL SOUNDING ARTIFICIAL REVERBERATION

DELAY FEEDBACK IN LOOP

Schroeder [1] descrive i principi ed il funzionamento di un filtro *comb IIR* consistente in un ritardo di n campioni in feedback. L'implementazione digitale diretta presenta un errore di offset di 1 campione in eccedenza rispetto alla risposta all'impulso documentata dall'autore.

```
//-------------------------- DELAY FEEDBACK IN LOOP ---
//
dfl(t,g) = (+ : @(t))~*(g);
```

Lo slittamento di 1 campione è dovuto all'implicito ritardo accumulato nell'istanza di feedback. Nell'implementazione seguente l'errore viene risolto con una semplice sottrazione di 1 campione internamente al loop e l'aggiunta di un campione di ritardo prima dell'uscita.

```
//------------------ DELAY FEEDBACK IN LOOP - FIXED ---
//
dflc(t,g) = (+ : @(t-1))~*(g) : mem;
```

La soluzione successiva si differenzia dalla precedente attraverso la possibilità di variare il tempo di ritardo, allocando un limite di memoria.

```
//-------------------------- DFL - FIXED - VARIABLE ---
//
dflcc(t,g) = (+ :
          de.delay(ma.SR/2, int(t-1)))~*(g) : mem;
```

ALL-PASS FILTER

L'implementazione di Schroeder del filtro *all-pass* prevede il riutilizzo del circuito di feedback del filtro *comb* precedentemente descritto.

```
//-------------------------------- ALL-PASS FILTER ---
//
apf(t,g) = _ <: *(-g)+(dflcc(t,g)*(1-(g*g)));
```

T60

Riferendosi alle ricerche di Sabine, Schroeder calcola con questa formula il tempo di decadimento del filtro riverberante fino ad una attenuazione di $60dB$, attraverso l'utilizzo delle variabili t tempo di ritardo e g coefficiente di attenuazione del circuito di feedback. La relazione tra i due esprime la perdita di intensità del circuito ad ogni rivoluzione.

```
//-------------------------------------------- T60 ---
// For a feedback loop with open loop gain g, the sound
// level decay by -20*log(g) decibels for every trip
// around the feedback loop. Since every round trip
// takes t second, the time for a 60dB decay is
//
msT60(t,g) = (60/(-20*log10(g)))*t;
process = msT60(0.1,0.708); // 2 seconds
```

ALL-PASS REVERBERATOR

Il tempo di ritardo è calcolato con la proporzione di 1/3 rispetto al ritardo precedente, opportunamente scalato al numero primo più prossimo.

```
//------------------------ INCREASE OF ECHO DENSITY ---
// The delay of each section is made about 1/3 of the
// preceding delay. Thus, the delay of the n-th unit
// will be t1(1/3)^n-1. The gains are most conveniently
// made equal to about 0.7. [...] The effective echo
// density of 5 loops in series will be approximately
// 81/t1. For t1 = 0.1 sec, the effective echo density
// will be 810 per second which is sufficiently close
// to the required 1000 per second. To avoid echo
```

```
// cancellation and superposition, it is advisable to
// use incommensurate delay ratios rather than the
// round number 3.
//
ied5 = apf(5507,0.7) :
     apf(1831,0.7) :
     apf(613,0.7) :
     apf(199,0.7) :
     apf(67,0.7);
```

NON-EXPONENTIAL DECAY

Nella seguente implementazione viene realizzato un riverbero con *decay* lineare.

```
//--------- MIXING OF DIRECTSOUND AND REVERBERATION ---
//--------------------------- NON-EXPONENTIAL DECAY ---
//
dflr(t,g) = (+ : de.delay(ma.SR/2, int(t-1)) :
          ied5)~*(g) : mem;
aprdw(t,g) = _ <: *(-g)+(dflr(t,g)*(1-(g*g)));
```

COMB FILTER REVERBERATION

Per una riverberazione naturale, Schoreder ipotizza l'utilizzo di 4 filtri *comb* in parallelo, sommati ed elaborati da due filtri *all-pass* in serie. Il valore di ritardo dei *comb* deve essere compreso, attraverso l'uso di numeri primi, tra i 30 e 45ms. Di seguito viene dimostrato come l'inversione della disposizione di *comb* e *all-pass* restituisca lo stesso risultato.

```
//------------------------ THE COMB FILTER APPROACH ---
// The values of the delays t1 through t4 are spread
// [...] between 30 and 45 ms here in prime numbers at
// 48KHz from 1440 to 2160 samples.
//
schrev = _ <: // to 4 parallel comb
  dflc(1447,0.812),
  dflc(1721,0.78),
  dflc(1873,0.76),
  dflc(2161,0.74) :>
  apf(83,0.7) : apf(229,0.7);
//
// components in reverse order
// two allpass to 4 comb
//
revsch = apf(83,0.7) : apf(229,0.7) <:
  dflc(1447,0.812),
  dflc(1721,0.78),
  dflc(1873,0.76),
  dflc(2161,0.74);
// commutability test
//
process = os.impulse <: schrev, (revsch :> _);
```

FREQUENCY-DEPENDENT REVERBERATION TIME

Nonostante la letteratura abbia attribuito a Moorer l'idea del filtraggio del segnale riverberato al fine di un decadimento più naturale, in realtà Schroeder lo ipotizza già nel suo articolo del 1962. Di seguito l'implementazione dello stesso.

```
//----------------- FREQUENCY-DEPENDENT REVERB TIME ---
// If it is desired to make the reverberation time a
// function of frequency, the gains g1 through g4 in
// have to be made frequency dependent. For this, a
// simple RC-section in each feedback loop will
// suffice. In this manner further realism can be added
// to the artificial reverberation by making the
// reverberation time larger for the low frequencies.
// This trend of reverberation time with frequency is
// typical of many concert halls and cathedrals.
//
dflf(t,g,fc) = (+ : de.delay(ma.SR/2,
          int(t-1)))~scy.onepole(fc)*(g) :
          mem;
//
schfdrevt = _ <: // to 4 parallel comb
  dflf(1447,0.812,5000),
  dflf(1721,0.78,4000),
  dflf(1873,0.76,3000),
  dflf(2161,0.74,2000) :>
  apf(83,0.7) : apf(229,0.7);
```

Il filtro utilizzato è il successivo, si tratta di un filtro *onepole* realizzato a partire da quello implementato da *Cycling74'* in *maxmsp*.

```
//-------------------------------- max-msp onepole~ ---
clip(a,b) = max(a) : min(b);
onepole(fc) = *(a) : +~*(ac)
with{
   a= sin(abs(fc)*2*ma.PI/ma.SR) : clip(0,1);
   ac = 1-a;
};
```

NEXT PRIME

Per una riveberazione complessa, senza fenomeni ripetitivi e sincroni si rivela necessario l'utilizzo di numeri primi che impedisce il propagarsi di ritardi multipli e sottomultipli. Di seguito viene presentata una funzione in linguaggio *c*, che, opportunamente importata su *Faust*, permette di prelevare dal tempo di ritardo immesso in campioni il numero primo successivo.

```
#include <stdio.h>
#include <math.h>

int is_prime(int num);
int next_pr(int num);

int next_pr(int num){
  int c;
  if(num < 2)
    c = 2;
  else if (num == 2)
    c = 3;
  else if(num & 1){
    num += 2;
    c = is_prime(num) ? num : next_pr(num);
  } else
    c = next_pr(num-1);

    return c;
}

int is_prime(int num){
  if((num & 1)==0)
    return num == 2;
```

```
  else {
    int i, limit = sqrt(num);
    for (i = 3; i <= limit; i+=2){
    if (num % i == 0)
    return 0;
    }
  }
  return 1;
}
```

La riga di import della funzione esterna in Faust che riporta il numero in uscita dalla funzione *next_pr*.

```
np = ffunction(int next_pr(int), <nextprime.h>,"");
```

AMBIOPHONIC REVERBERATION

Il riverbero *ambiophonic* consiste nella diffusione del riverbero digitale mediante molteplici linee di segnale, attraverso altrettanti elementi di diffusione acustica. Schroeder identifica in questa tecnica di riverberazione la possibilità di correzione dello spazio acustico di una sala da concerto oltre che della sua aumentazione in termini di possibili caratteristiche riverberanti.

Lo sviluppo di questa parte della ricerca è appena iniziato. Il progetto prevede l'utilizzo di sedici linee di riverberazione con controllo della frequenza di taglio dei filtri *all-pass* a valle della catena, in modo da poter controllare attivamente ed adattivamente fino a 16 frequenze di risonanza dell'ambiente acustico.

> This procedure, which is already obsolete, makes use of delaying devices reproducing not only the discrete initial reflections; but also the reverberation tail. The reflection sequences have herewith to be chosen in such a way that no comb-filter effects, such as flutter echoes, will be produced with impulsive music motifs. The functioning of a simple ambiophonic system can be described as follows: to the direct sound emanating directly from the original source and directly irradiated into the room, there are admixed delayed signals produced by an adequate sound delaying system (in the initial stages this was just a magnetic sound recording system) which are then irradiated like reflections arriving with corresponding delay from the walls or the ceiling. This requires additional loudspeakers appropriately distributed in the room for irradiating the delayed sound as diffusely as possible. For further delaying the sound it is possible to arrange an additional feedback from the last output of the delay chain to the input. A system of this kind was first suggested by Kleis51 and was installed realized in several large halls. 52, 53 [6]

> In 1965, the thirty-four-year-old studio received a much-deserved facelift, including the installment of a state-of-the- art ambiophonics system. The brainchild of Gilbert Dutton, the head of EMI's research labs, the ambiophonics system was designed to increase Studio 1's short reverb time, which typically clocked in around two seconds. The idea behind Dutton's innovation was to afford the spacious studio with the sound and feel of a concert hall. The ambiophonic process, as Dutton devised it, was relatively simple. The microphones in the studio sent a slightly delayed signal that would be played back through a series of loudspeakers installed on the walls of the mammoth room. The new signal would be picked up, in turn, by the original set of microphones and recorded. In Dutton's design, an increase in reverb would be realized by virtue of the length of the delay and the distance between the speakers and the mics. Dutton's ambiophonic system required ninety-six loudspeakers in order to create the necessary sound diffusion. Historian Howard Massey has described ambiophonics as "the Grand Experiment that never quite worked." And in truth, the whole apparatus fell somewhat short of Dutton's original ambition for creating a kind of midcentury forerunner of contemporary surround sound. Given the limitations of 1960s-era technology, the system maxed out after six signal delays—a process that was made possible by the installation of a "delay drum," which consisted of a rotating metal platter, its outer edge having been treated with ferric oxide, with seven magnetic heads (one for recording, six for playback) randomly interspersed around its perimeter. Each of the playback heads directed its signal to a preamplifier, which returned the signal to sixteen of the loudspeakers installed on the walls of Studio 1. As Massey explained, "The whole system was essentially a large feedback loop, and therein lay the rub: It only functioned best when on the verge of howling, which made it largely uncontrollable." For his part, Townsend concurred, feeling that ambiophonics "was too artificial. The results sounded a little phony." In many ways, Dutton's system was just another one of the several discrete elements that needed to come together on that magical evening to bring George and the Beatles' vision for "A Day in The Life" into reality.

ALGORITMO DI STUDIO

```
import("stdfaust.lib");
// ------ comb
//
dflc(t, g) = (+ : de.delay(ma.SR, int(t-1)))
          ~*(max(0, min(0.999, g))) : mem;
// ------ allpass
//
alp(t, g) = _<:(_*(ma.neg(max(0.5, min(0.9, g))))) +
            (dflc(t, g)*(1-(g*g)));
// ------ filtro passa basso "onepole" Dattorro
     inspired
//
clip(a,b) = max(a) : min(b);
onepole(fc) = *(a) : +~*(ac)
with{
   a= sin(abs(fc)*2*ma.PI/ma.SR) : clip(0,1);
   ac = 1-a;
};
// ------ importa funzione nextprimes.h in faust
//
//np = ffunction(int next_pr(int), <nextprime.h>,"");
np = *(1);
// ------ sliders feedback
```

```
//
g(x) = hgroup("[01]CONTROL",x);
  gr(x) = g(hgroup("feedback", x));
     fbkc = gr(hslider("[02]DIFFUSION [style:knob]
          ", 0.708, 0, 0.9, 0.01));
     fbk = gr(hslider("[03]FB FDN [style:knob]
          ", 0.708, 0, 0.99, 0.01) : si.smoo) ;
     fbka = gr(hslider("[01]INPUT AP [style:knob]
          ", 0.708, 0, 0.99, 0.01) : si.smoo);
     fbkm = gr(hslider("[04]MATRIX [style:knob]
          ", 0.708, 0, 0.99, 0.01) : si.smoo);
// ------ slider filtro
//
glp(x) = g(hgroup("filtro",x));
  bandwidth = glp(hslider("BANDWIDTH [style:knob]
           ", 0.9995, 0.0001, 0.9999999, 0.0000001));
  damping = glp(hslider("DAMPING [style:knob]
           ", 0.1, 0, 0.999, 0.001));
// ------ sliders posizione
//
xg(x) = hgroup("[02]ROOM", x);
  yg(x) = xg(hgroup("vertical", x));
  size = xg((hslider("SIZE[style:knob]",1,0,2,0.01)));
  x = ((size*(7)));
  y = ((size*(5)));
  z = ((size*(3)));
  xps = xg(hslider("Position",0.5,0,1,0.01));
// ------ sliders vari
//
dist = vslider("distance", 1,0.01,20,0.01);
primes = xg((hslider("PRIMES [style:knob]
       ",2,2,24000,1)));
// ------ position values limitati ai valori x,y,z
//
xp = x*xps : np;
// ------ n allpass in serie + offset L/R
//
primeseq(n) = seq(i,n,np);
// ------ route + feedback delay network
//
ambifdn4(nalp,n) = (xroute :
   ((alpar(x-xp,fbkc,nalp))*(xps)),
   ((alpar(xp,fbkc,nalp))*(1-xps)),
   (alpar(y,fbkc,nalp)),
   (alpar(z,fbkc,nalp)))
   ~ (xfbk(n) : feedbackmatrix(n)) : rut
with {

  xroute(a,b,c,d,a1,b1,c1,d1)=a+a1,b+b1,c+c1,d+d1;
  feedbackmatrix(N) = bhadamard(N);
  bhadamard(2) = si.bus(2) <: +,-;
  bhadamard(n) = si.bus(n) <: (si.bus(n):>si.bus(n/2)),
     ((si.bus(n/2),(si.bus(n/2):par(i,n/2,*(-1)))) :>
     si.bus(n/2)) : (bhadamard(n/2) , bhadamard(n/2));
  xfbk(n) = par(i,n,onepole(damping)*(fbk)*(1/sqrt(n)))
       ;
  alpar(sb, fb, N)=seq(i,N,alp(pm.l2s(sb:primeseq(i+1)),
       fb));
  rut(l, r ,y, z)=l+((y+z)*0.5),r+((y+z)*0.5);
  };
// ------ Reverb systems
//
%-------------------------------------------------------
ambidat2ch(nch) = onepole(bandwidth) <:
                ambifdn4(nch,4);
// ------ process
//
process = ambidat2ch(4);
```

References

[1] Manfred Robert Schroeder. Natural sounding artificial reverberation. *Journal of the Audio Engineering Society*, page 5, 1962.

[2] Vesa Välimäki, Julian Parker, Lauri Savioja, Julius Smith, and Jonathan Abel. Fifty years of artificial reverberation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20:1421 –1448, 07 2012.

[3] Anders Torger and Angelo Farina. Real-time partitioned convolution for ambiophonics surround sound. pages 195 – 198, 02 2001.

[4] Ralph Glasgal. Surround ambiophonic recording and reproduction. 07 2003.

[5] Kenneth Womack. *Sound pictures, the life of Beatles producer George Martin*. 2018.

[6] Glen Ballou. *Handbook for sound engineers*. Focal Press, 2008.