FDN & Spectral Processing - eine Projektarbeit im Rahmen der Veranstaltung: Signalverarbeitung und Audiotechnik

Simon Johannes Braun
Matrikelnummer: 091603717

1 Feedback Network Delay

Im Seminar Signalverarbeitung im Wintersemester 19/20 haben wir in Gruppenarbeit und als gemeinsames Projekt ein Feedback Delay Network entwickelt. Nach gemeinsamen konzeptionellen Gesprächen, übernahmen wir, die Seminarteilnehmer, die eigenständige Entwicklung einzelner Komponenten, die anschließend zu dem Endergebnis zusammengeführt wurden.

Das Feedback Delay Network besteht aus mehreren eigenständigen Klangerzeugern und der Aufnahme und Verarbeitung von Umgebungsgeräuschen. Das aufgenommene Audiosignal wie auch die intern generierten Audiosignale werden mit Hilfe der "Fast Fourier Transformation" in einzelne Frequenzspektren aufgeteilt. Diese einzelnen analysierten Frequenzen werden danach wiederum an die internen Klangerzeuger übergeben, die mit diesen Frequenzen neue Audiosignale erzeugen. Dieser Prozess wiederholt sich in zufälligen Abständen in einer Endlosschleife. Dadurch wird stetig neuer Raumklang erzeugt, der gleichzeitig wieder aufgenommen, analysiert, re-synthetisiert und wiedergegeben wird.

2 Fast Fourier Transformation

Die Fast Fourier Transformation (FFT) ist ein optimierter Algorithmus zur Implementierung der "Diskreten Fourier Transformation"[1]. Dabei wird ein bestimmter und begrenzter Anteil eines Signals in seine Bestandteile zu einzelnen Sinusschwingungen zerlegt. Deren jeweilige Amplituden und Phasen können dann bestimmt werden. Um ein Signal auf werte in einem bestimmten Frequenzbereich zu untersuchen, ist für die FFT die geeignete Abtastrate und Abtastwerte (Samples) zu bestimmen. Der Wert der Abtastrate ist mindesten doppelt so groß zu wählen, wie die höchste zu analysierende Frequenz sein soll. Also legt der halbe Wert der Abtastrate die Obergrenze für das Frequenzspektrum fest, mit dem das Signal untersucht werden kann. Die Samples bestimmen die Auflösung der einzelnen Amplituden-Messwerte. Je mehr Samples, desto genauer aber auch aufwendiger die Berechnung einzelner Amplituden zu bestimmten Zeitpunkten. Bei einer Abtastrate von 48000 Hz und einer Samplerate von 1024 `(2^10)` können so Frequenzunterschiede in 46.88 Hz Schritten `(48000 Hz/1024 )` über eine Messdauer von 21.33 ms `(1024/48000 H)` ermittelt werden. Da ein bestimmter Bestandteil eines Signals analysiert werden kann, ist es wahrscheinlich, dass die Amplituden am Anfang und am Ende des Signalausschnitts nicht null sind.[2] Das führt bei der FFT zu unerwünschten Fehl-

[1] McGee, Kevin J., An Introduction to Signal Processing and Fast Fourier Transform, [online] http://www.fftguru.com/fftguru.com.tutorial.pdf [14.01.2020]

[2] National Instruments, Understanding FFTs and Windowing, [online] https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf [14.01.2020]

interpretationen, die sich in Frequenzen bemerkbar machen, die im Ursprungssignal gar nicht vorhanden sind. Um dem entgegenzuwirken, wird die Methode des Fensterns (windowing) angewandt. Dabei wird der Signalausschnitts am Anfang und am Ende mit null multipliziert. (z.B. Hamming).

3 drone.csd

```
<CsoundSynthesizer>
<CsInstruments>
instr 3

ishift      =           .00666667                           ;shift it 8/1200.
ipch        =           cpspch(p5)                          ;convert parameter 5 to cps.
ioct        =           octpch(p5)                          ;convert parameter 5 to oct.
kadsr       linseg      0, p3/3, 1.0, p3/3, 1.0, p3/3, 0    ;ADSR envelope
kmodi       linseg      0, p3/3, 5, p3/3, 3, p3/3, 0        ;ADSR envelope for I
kmodr       linseg      p6, p3, p7                          ;r moves from p6->p7 in p3 sec.
a1          =           kmodi*(kmodr-1/kmodr)/2
a1ndx       =           abs(a1*2/20)                        ;a1*2 is normalized from 0-1.
a2          =           kmodi*(kmodr+1/kmodr)/2
a3          tablei      a1ndx, 23, 1                        ;lookup tbl in f23, normal index
ao1         oscil       a1, ipch, 22                        ;cosine
a4          =           exp(-0.5*a3+ao1)
ao2         oscil       a2*ipch, ipch, 22                   ;cosine
aoutl       oscil       0.05*kadsr*a4, ao2+cpsoct(ioct+ishift), 21 ;fnl outleft
aoutr       oscil       0.05*kadsr*a4, ao2+cpsoct(ioct-ishift), 21 ;fnl outright
            outs        Sigmoide(aoutl), Sigmoide(aoutr)-

endin
</CsInstruments>
</CsoundSynthesizer>
```

```
<CsoundSynthesizer>
<CsScore>
;f tables for instr 3
f21 0 8192 10 1                                 ;hughres sine wave
f22 0 8192 11 1                                 ;highres cosine wave
```

```
f23 0 8192 -12 20.0                                    ;unscaled ln(I(x)) from 0 to 20.0
</CsScore>
</CsoundSynthesizer>
```

```
<CsoundSynthesizer>
<CsScore>
;instr3    start    dur     gain    freq    amp    amp2
;p1        p2       p3      p4      p5      p6     p7


; p5 shift octaves
i3 0 30 1   4.06 1.0 0.2
i3 . . .    5.01 . .
i3 . . .    5.06 . .
i3 . . .    5.10 . .
i3 . . .    5.11 . .
i3 . . .    6.04 . .

i3 10 15 1  5.06 0.8 0.2
i3 . . .    6.01 . .
i3 . . .    6.06 . .
i3 . . .    6.10 . .
i3 . . .    6.11 . .
i3 . . .    7.04 . .

i3 20 80 1  4.06 1.0 0.2
i3 . . .    5.01 . .
i3 . . .    4.06 . .
i3 . . .    4.10 . .
i3 . . .    4.11 . .
i3 . . .    4.04 . .
</CsScore>
</CsoundSynthesizer>
```

```
<CsoundSynthesizer>
<CsOptions>
```

```
-o dac
;-i dac
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
nchnls = 2
0dbfs = 1


gipt init 0  ;global playtime

gkfreg init 0
gkrate init 0
gkbandw init 0

;gkfreq invalue 'freq'
;gkrate invalue 'rate'
;gkbandw invalue 'bandw'

opcode Sigmoide, a, a
    aX xin
    aPow = exp(aX*3.0)
    aOut = 2.0 * (1.0 / (1.0 + aPow))- 1.0
    xout aOut
endop


instr 1

anoise pinkish 1
kfreq = p4
kbandw = 10
alp butbp anoise*p5, kfreq, kbandw

outs alp, alp

endin

```

```
instr 3

ishift      =           .00666667                       ;shift it 8/1200.
ipch        =           cpspch(p5)                      ;convert parameter 5 to cps.
ioct        =           octpch(p5)                      ;convert parameter 5 to oct.
kadsr       linseg      0, p3/3, 1.0, p3/3, 1.0, p3/3, 0    ;ADSR envelope
kmodi       linseg      0, p3/3, 5, p3/3, 3, p3/3, 0        ;ADSR envelope for I
kmodr       linseg      p6, p3, p7                      ;r moves from p6->p7 in p3 sec.
a1          =           kmodi*(kmodr-1/kmodr)/2
a1ndx       =           abs(a1*2/20)                    ;a1*2 is normalized from 0-1.
a2          =           kmodi*(kmodr+1/kmodr)/2
a3          tablei      a1ndx, 23, 1                    ;lookup tbl in f23, normal index
ao1         oscil       a1, ipch, 22                    ;cosine
a4          =           exp(-0.5*a3+ao1)
ao2         oscil       a2*ipch, ipch, 22               ;cosine
aoutl       oscil       p4*0.05*kadsr*a4, ao2+cpsoct(ioct+ishift), 21 ;fnl outleft
aoutr       oscil       p4*0.05*kadsr*a4, ao2+cpsoct(ioct-ishift), 21 ;fnl outright
            outs        Sigmoide(aoutl), Sigmoide(aoutr)

endin

instr 2

ifn       =         p7                        ;f table number

i1        =         p6                        ; INIT VALUES CORRESPOND TO FREQ.
i2        =         2*p6                      ; OFFSETS FOR OSCILLATORS BASED ON ORIGINAL p6
i3        =         3*p6
i4        =         4*p6

ampenv    linen     p5,30,p3,30               ; ENVELOPE


a1        oscili    ampenv,p4,ifn
a2        oscili    ampenv,p4+i1,ifn          ; NINE OSCILLATORS WITH THE SAME AMPENV
a3        oscili    ampenv,p4+i2,ifn          ; AND WAVEFORM, BUT SLIGHTLY DIFFERENT
a4        oscili    ampenv,p4+i3,ifn          ; FREQUENCIES TO CREATE THE BEATING EFFECT
a5        oscili    ampenv,p4+i4,ifn
a6        oscili    ampenv,p4-i1,ifn          ; p4 = fREQ OF FUNDAMENTAL (Hz)
a7        oscili    ampenv,p4-i2,ifn          ; p5 = AMP
```

```
a8        oscili    ampenv,p4-i3,ifn            ; p6 = INITIAL OFFSET OF FREQ - .03 Hz
a9        oscili    ampenv,p4-i4,ifn

;asnd      =         (a1+a2+a3+a4+a5+a6+a7+a8+a9)/9

asum      =         (a1+a3+a5+a7+a9+a2+a4+a6+a8)/p5

          ;outs     a1+a2+a3+a4,a5+a6+a7+a8+a9


          outs      asum, asum


;garvbsig  =         garvbsig+(asnd*.85)


endin

instr 4

ifn = 1                                       ;choose wave form table
ktrig metro 10
kGate     randomi   0, 1, 3
kdB       randomi   0, 12, 5


if kGate > 0.5 then                           ;if kGate is larger than 0.5
kVol      =         kdB/12                    ; open gate
 else
kVol      =         0                         ;otherwise close gate
 endif
kVol      port      kVol, .02                 ;smooth volume curve to avoid clicks

kamp linseg 0, 1, 0.2, 1, 0

aSig1 oscil3 kamp, 60, ifn
aSig2 oscil3 kamp, 120, ifn
aSig3 oscil3 kamp, 240, ifn
aSig4 oscil3 kamp, 320, ifn
aSig5 oscil3 kamp, 440, ifn
```

```
aSig6 oscil3 kVol, 60, ifn
aSig7 oscil3 kVol, 120, ifn

;aPhas phasor 300

;aSig2 vco2 0.5, 300

;aPhas poscil kVol,60,2

aSigSum = aSig1 + aSig2 + aSig3 + aSig4 + aSig5 + aSig6 + aSig7

arev reverb aSigSum, 2

outs arev, arev
endin


</CsInstruments>

<CsScore>

;f tables for instr 2
f1  0 512   9 1 1 0                                 ;sine lo-res
f2  0 512   5 4096 512 1                            ;exp env
f3  0 512   9 10 1 0 16 1.5 0 22 2 0 23 1.5 0       ;inharm wave
f4  0 512   9 1 1 0                                 ;sine
f8  0 512   5 256 512 1                             ;exp env
f9  0 512   5 1   512 1                             ;constant value of 1
f10 0 512   7 0 50 1 50 .5 300 .5 112 0             ;ADSR
f11 0 2048 10 1                                     ;SINE WAVE hi-res
f13 0 1024  7 0 256 1 256 0 256 -1 256 0            ;triangle
f14 0 512   7 1 17  1 0   0 495                     ;pulse for S&H clk osc
f15 0 512   7 0 512 1 0                             ;ramp up;;;left=>right
f16 0 512   7 1 512 0 0                             ;ramp down;;;right=>left
f17 0 1024  7 .5 256 1 256 .5 256 0 256 .5          ;triangle with offset
f18 0 512   5 1 512 256                             ;reverse exp env
f20 0 1024 10 1 0 0 0 .7 .7 .7 .7 .7 .7             ;approaching square

;f tables for instr 3
```

```
f21 0 8192 10 1                                    ;hughres sine wave
f22 0 8192 11 1                                    ;highres cosine wave
f23 0 8192 -12 20.0                                ;unscaled ln(I(x)) from 0 to 20.0



;instr    start     dur     freq    amp     offset      table number
;p1       p2        p3      p4      p5      p6          p7
;i1       0         100     120     10
;i2       40        40      33      10     .03          20
;i4       0         100

;instr3    start    dur     gain    freq    amp    amp2
;p1       p2        p3      p4      p5      p6     p7


; p5 shift octaves
i3 0 30 1   4.06 1.0 0.2
i3 . . .    5.01 . .
i3 . . .    5.06 . .
i3 . . .    5.10 . .
i3 . . .    5.11 . .
i3 . . .    6.04 . .

i3 10 15 1  5.06 0.8 0.2
i3 . . .    6.01 . .
i3 . . .    6.06 . .
i3 . . .    6.10 . .
i3 . . .    6.11 . .
i3 . . .    7.04 . .

i3 20 80 1  4.06 1.0 0.2
i3 . . .    5.01 . .
i3 . . .    4.06 . .
i3 . . .    4.10 . .
i3 . . .    4.11 . .
i3 . . .    4.04 . .
```

```
i3 5 30 1   4.16 1.0 0.2
i3 . . .    5.11 . .
i3 . . .    5.16 . .
i3 . . .    5.20 . .
i3 . . .    5.21 . .
i3 . . .    6.14 . .

i3 15 15 1  5.16 0.8 0.2
i3 . . .    6.11 . .
i3 . . .    6.16 . .
i3 . . .    6.20 . .
i3 . . .    6.21 . .
i3 . . .    7.14 . .

i3 35 15 1  4.16 1.0 0.2
i3 . . .    5.16 . .
i3 . . .    5.11 . .
i3 . . .    5.20 . .
i3 . . .    5.21 . .
i3 . . .    5.14 . .



i3 15 30 1  4.03 1.0 0.2
i3 . . .    4.01 . .
i3 . . .    4.06 . .
i3 . . .    4.00 . .
i3 . . .    4.01 . .
i3 . . .    6.04 . .

i3 25 15 1  4.06 0.8 0.2
i3 . . .    4.11 . .
i3 . . .    4.16 . .
i3 . . .    4.20 . .
i3 . . .    4.21 . .
i3 . . .    4.14 . .

i3 45 15 1  4.06 1.0 0.2
i3 . . .    5.11 . .
i3 . . .    4.16 . .
```

```
i3 . . .    4.20 . .
i3 . . .    4.21 . .
i3 . . .    4.14 . .
</CsScore>

</CsoundSynthesizer>
```