UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria dell'Informazione

TESI DI LAUREA TRIENNALE
IN
INGEGNERIA DELL'INFORMAZIONE

Realizzazione in tempo reale di filtri IIR risonanti in un modello di filtraggio acustico dell'orecchio per il rendering spaziale del suono

Relatore: *Prof. Federico Avanzini*
Correlatore: *Michele Geronazzo*

Laureando: Precious Ugo Abara

ANNO ACCADEMICO 2011 / 2012

Indice generale

Sommario.....5

1 Il Suono.....7

1.1 Proprietà fisiche del suono.....7
1.2 Percezione del suono.....8
1.2.1 La testa.....8
1.2.2 Orecchio esterno.....11
1.3 Rendering Spaziale.....14
1.4 Head-Related Transfer Function (HRTF) 17
1.4.1 Modello strutturale.....18
1.4.2 Modello per la pinna.....19

2 PureData.....21

2.1 Panoramica.....21
2.1.1 PD-Extended.....22
2.2 PD window e patch window 22
2.3 Il motore DSP.....24
2.4 Le scatole di PD.....24
2.4.1 Oggetti e connessioni.....25
2.4.2 print 25
2.4.3 Oggetti più comuni 26
2.4.4 Esempio di patch PD.....26
2.5 Segnali audio.....27
2.5.1 L'oggetto dac~.....28
2.6 Internal, External e Librerie.....29

3 Realizzazione filtri.....31

3.1 Componente risonante.....32
3.2 Componente riflessa.....33
3.3 Filtri.....36
3.4 Implementazione filtro in PD.....39
3.5 Risultati.....46

4 Conclusioni.....49

Appendice.....51

A.1 hres~.....51
A.1.1 Filtro hres di tipo 1 (hres~ 1).....51
A.1.2 Filtro hres di tipo 2 (hres~ 2).....53
A.1.3 Calcolo output filtro hres~.....55
A.1.4 Funzione interpolazione.....56

A.1.5 Creazione di hres~ in PD.....59
A.1.6 Oggetto Peak.....62
A. 2 hrefl~.....64
A.2.1 Filtro Hrefl~.....64
A.2.2 Constanti e variabili importanti.....66
Bibliografia 67

SOMMARIO

Il progetto presentato in questa tesi si inserisce all'interno di un progetto più ampio per il rendering spaziale del suono, tramite tecniche binaurale, e consiste nell'implementazione di due *filtri* di sintesi delle PRTF (Pinna-Related Transfer Function) in Pure Data: un'ambiente di elaborazione digitale audio e video in tempo reale. Le PRTF sono delle funzioni che indicano il contributo della pinna nelle HRTF(Head-Related Transfer Function). Le HRTF hanno il compito di "catturare" le trasformazioni subite da un'onda sonora nel tragitto dalla sorgente ai timpani, dovute ai fenomeni di diffrazione e riflessione del corpo dell'ascoltatore(testa, torso, pinna e spalle). Poiché sono un insieme di diversi contributi molti complessi da valutare a livello globale, si cerca un'astrazione arrivando al *modello strutturale.*

Capitolo 1: Introduzione al suono, le sue proprietà fisiche come onda e l'introduzione ai concetto di suono 3D, di spazializzazione e percezione del suono. Verranno analizzati i modelli matematici usati per modellare, tramite tecniche binaurali, lo spostamento di una sorgente nello spazio. Verrà inoltre analizzata la funzione di ostacolo del corpo nel ascoltatore e le sue conseguenze sulle trasformazioni subite dall'onda sonora.

Capitolo 2: Una breve introduzione all'ambiente di elaborazione digitale di segnale audio e video Pure Data. Verrà data una breve panoramica delle funzioni principali e delle nozioni di patch, di internal e di external, che sono indispensabili per lo svolgimento del progetto presentato in questa tesi.

Capitolo 3: Qui verranno esposte in maniera dettagliata le external e le patch realizzate. Saranno infatti analizzate tutte le funzioni implementate, con la relativa descrizione matematica, e tutte le parti della patch realizzata. Infine si fornirà un esempio di utilizzo della stessa con relative valutazioni sul risultato ottenuto.

Capitolo 4: In questo capitolo conclusivo saranno messi in evidenza i limiti e problemi del modello presentato. Verrà data una breve panoramica sulla ricerca allo stato dell'arte riguardante il modello.

1 IL SUONO

1.1 PROPRIETÀ FISICHE DEL SUONO

Il suono è una vibrazione meccanica (oscillazione) che si propaga in un mezzo elastico. Quest'ultima provoca dei movimenti nello spazio, intorno alla posizione di riposo e lungo la direzione di propagazione, di particelle adiacenti alla sorgente del suono. La vibrazione si propaga meccanicamente dando origine a un'onda sonora, che è pertanto onda longitudinale. Le onde longitudinali hanno la capacità di attraversare metalli e altri materiali solidi più velocemente rispetto alle onde trasversali.

Una grandezza fisica la cui evoluzione nel tempo e nello spazio sia rappresentata da una funzione che soddisfa l'equazione di d'Alembert è un'onda. L'equazione di d'Alembert è lineare e per essa vale il principio di sovrapposizione. Il suono, in quanto onda, soddisfa quindi l'equazione di d`Alembert.

$$\nabla^2 p(\boldsymbol{x},t)=\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}(\boldsymbol{x},t)$$

dove $\boldsymbol{x}$ rappresenta le coordinate Euclidee nello spazio, p è la pressione acustica, c è la velocità del suono nel mezzo elastico, t *è* la costante di tempo e $\nabla^2=\partial^2/\partial x^2+\partial^2/\partial y^2+\partial^2/\partial z^2$ è l'operatore laplaciano in tre dimensioni.

Qual è la velocità di propagazione di un'onda sonora? Dipende dalle proprietà del mezzo (ad esempio, il suono si propaga più velocemente nell'acqua che non nell'aria). Se la temperatura dell'aria è di circa 20°C, un'onda sonora viaggia ad una velocità di 343,8 m/s: leggermente più veloce a temperature più alte e più lenta a temperature più basse. Dal momento che la velocità del suono è inversamente proporzionale alla radice quadrata della massa molecolare, nell'elio si ha una velocità che è circa tre volte di quella nell'aria.

Il suono che si propaga nell'aria causa la vibrazione dell'orecchio in modo simile a quella della sorgente del suono. L'intensità del suono che investe un ricevitore (ad

esempio un microfono, o un orecchio) è inversamente proporzionale alla distanza dalla sorgente poiché l'area del fronte d'onda sferico aumenta con il quadrato della distanza (se *r* è la distanza, Area = $4\pi r^2$) e la potenza del suono è distribuita su tutta la superficie sferica. Le caratteristiche del suono e quelle spaziali vengono ulteriormente modificate da fenomeni di riflessione, interferenza e diffrazione. In una stanza, la maggior parte del suono che sentiamo proviene da riflessioni sul pavimento, soffitto e muri. Una sequenza di piccole riflessioni che arrivano a intervalli irregolari e che si estinguono dolcemente costituiscono la *riverberazione*, senza la quale il suono risulta piatto, secco e debole.

1.2 Percezione del suono

L'abilità di localizzare la sorgente di un determinato suono è di importanza vitale. La riverberazione svolge un ruolo fondamentale nell'individuare alcune caratteristiche della stanza: dimensioni, tipo di materiale delle pareti, dove è localizzata la sorgente e cosi` via. Ci fornisce anche dei suggerimenti sulla distanza e direzione dalle sorgente, e un senso di *immersione* nel suono.

1.2.1 La testa

Le nostre orecchie non sono oggetti isolati nello spazio. Sono posizionate alla stessa altezza agli antipodi di un oggetto ad alta impedenza acustica: la testa. Questa costituisce un ostacolo per la libera propagazione del suono ed ha due principali effetti:

- ITD (Interaural Time Difference), causato dal fatto che le onde sonore devono percorre una maggiore distanza per raggiungere l'orecchio più lontano. Uno studio approssimato ma abbastanza preciso di questo fenomeno si può effettuare facendo delle ipotesi semplificative: si suppongono o la sorgente lontana e la testa sferica. La prima assunzione implica che le onde sonore che colpiscono la testa sono onde piane; la distanza ulteriore Δx, che un raggio sonoro dovrà percorrere per raggiungere l'orecchio più lontano, viene stima-

ta da considerazioni geometriche elementari (vedi Figura 1: Stima di ITD nel caso di sorgente lontana (onde piane) e testa sferica):

$$ITD \sim \frac{a}{c}(\theta + \sin(\theta))$$

Dove *c* e` la velocità del suono, *a* rappresenta il raggio della testa (tipicamente a = 8, 5cm) e θ l'angolo di azimuth 2 , che definiscono la direzione della sorgente sonora sul piano orizzontale; la formula mostra che il minimo ITD si ha quando la sorgente è frontale e (θ = 0), mentre è massimo quando la sorgente è ai lati della testa (θ = π/2).

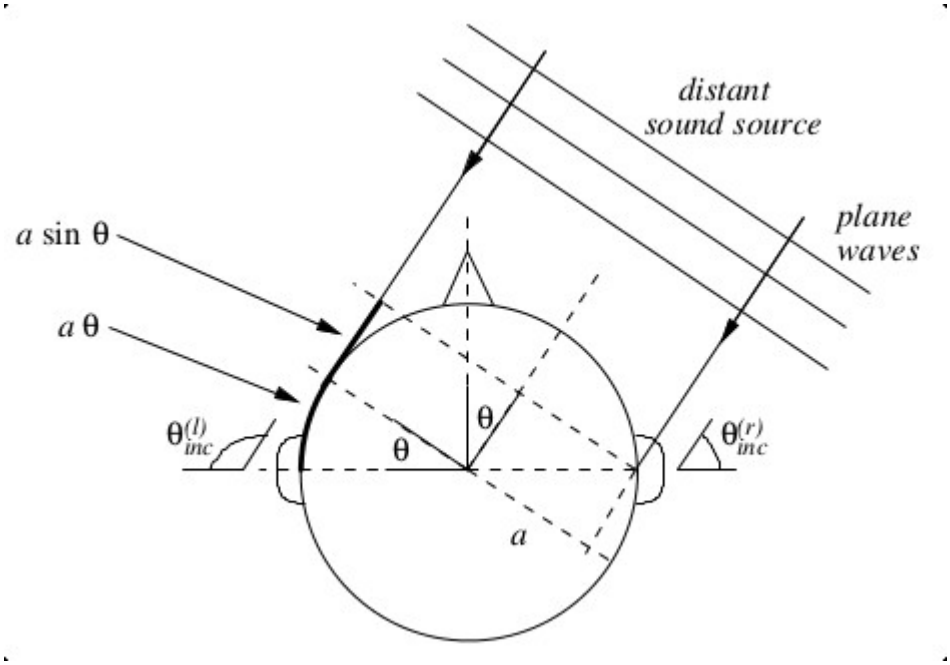

Figura 1: *Stima di ITD nel caso di sorgente lontana (onde piane) e testa sferica*

- ILD (Interaural Level Difference), poiché all'orecchio più lontano giunge un suono di minore intensità a causa della testa che introduce un'attenuazione. Mentre è accettabile supporre l'ITD come un parametro dipendente dalla frequenza, l'ILD è invece fortemente legato alla frequenza: a basse frequenze (per lunghezze d'onda dell'ordine del diametro della testa) non vi è quasi alcuna differenza, mentre ad alte frequenze diventa particolarmente significati-

vo.

Come già fatto in precedenza, l'ILD può essere analizzato nel caso ideale di testa perfettamente sferica (di raggio *a*) e che la sorgente sonora sia situata ad una distanza *r* > *a* dal centro della sfera.

È solito utilizzare costanti normalizzate: $\mu = \omega a/c$ (frequenza normalizzata) e $\rho = r/a$ (distanza normalizzata). Se si considera un punto sulla sfera allora la diffrazione di un'onda acustica causata dalla sfera nel punto scelto si può esprimere con la funzione di trasferimento:

$$H_{sphere}(\rho, \theta_{inc}, \mu) = -\frac{\rho}{\mu} e^{-i\mu\rho} \sum_{m=1}^{m=+\infty} (2m+1) P_m(cos\theta_{inc}) \frac{h_m(\mu\rho)}{h'_m(\mu)}$$

dove P_m e h_m sono, rispettivamente, i polinomi di Legendre di ordine m-esimo e la funzione sferica di Hankel; θ_{inc} rappresenta l'angolo di incidenza.

È noto che la funzione di Hankel $h_m(x)$ ammette un'approssimazione asintotica quando l'argomento, *x*, tende all'infinito.

Dal grafico in Figura 2 (Modulo di una sfera Hsphere(∞, θinc, μ) nel caso di una sorgente a distanza infinita) si nota che a bassa frequenza la funzione di trasferimento non è dipendente dalla direzione e il suo modulo, $| H_{sphere}(\rho, \theta_{inc}, \mu) |$, è pressoché costante ed uguale a 1 qualunque sia l'angolo. Se $\mu > 1$ la dipendenza dall'angolo di incidenza diventa molto sensibile.

È da notare inoltre che la risposta minima non si presenta per $\theta_{inc} = \pi$: questo è dovuto al fatto che tutte le onde sonore che si propagano attorno alla sfera arrivano in quel punto in fase; a frequenze elevate questo fenomeno diventa molto meno presente e la parte posteriore della sfera risulta essere in una zona di ombra sonora.

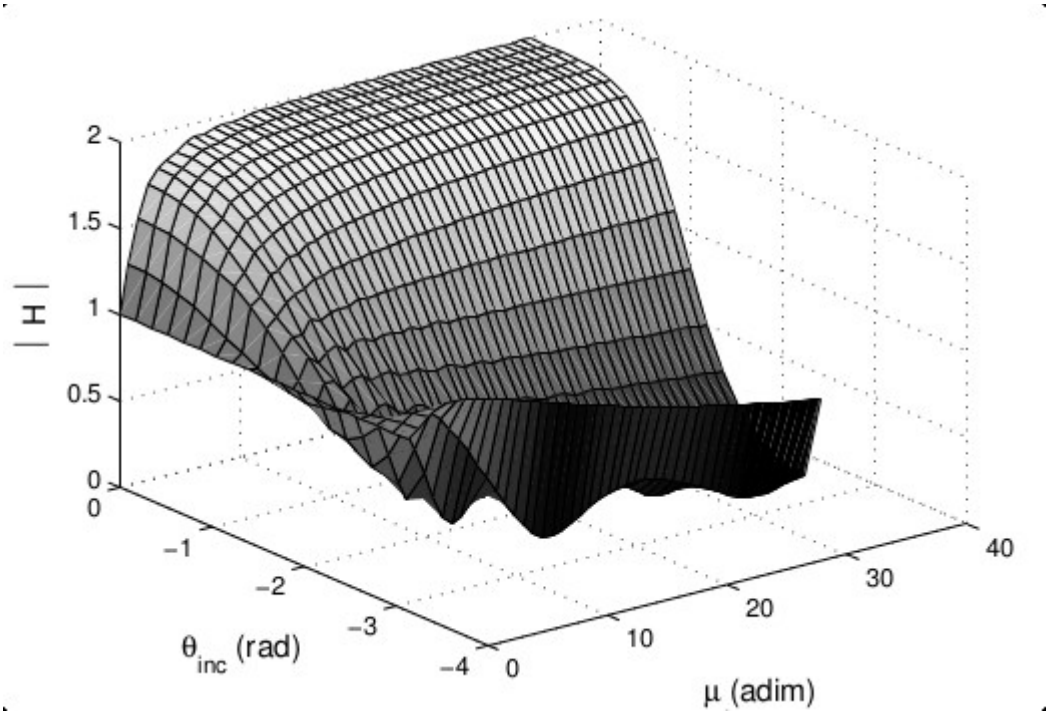

Figura 2: Modulo di una sfera $H_{sphere}(\infty, \theta_{inc}, \mu)$ nel caso di una sorgente a distanza infinita

1.2.2 ORECCHIO ESTERNO

La parte esterna dell'apparato uditivo della maggior parte dei mammiferi è piuttosto simile. L'orecchio esterno è composto dalla pinna(la parte effettivamente visibile) e dal condotto uditivo. La pinna è stata generalmente considerata una parte ininfluente dell'apparato uditivo per quanto riguarda la percezione del suono, ma, in realtà, modifica significativamente il suono in arrivo, specialmente alle alte frequenze e questo aiuta la nostra capacità di localizzare suoni. Il suono viaggia attraverso i condotti uditivi e provoca la vibrazione dei timpani. Le figure 3 (Orecchio esterno: pinna) e 4 (Orecchio esterno: condotto uditivo) mostrano la parte esterna di un orecchio.

La pinna ha dei bassorilievi caratteristici per ogni persona ed è connessa al condotto uditivo, descrivibile in prima approssimazione come un tubo di larghezza costante

con pareti ad alta impedenza acustica: esso si comporta come un risonatore monodimensionale, mentre la pinna svolge la funzione di antenna acustica. Le cavità di quest'ultima amplificano alcune frequenze, mentre la sua particolare geometria introduce interferenze che ne attenuano altre.

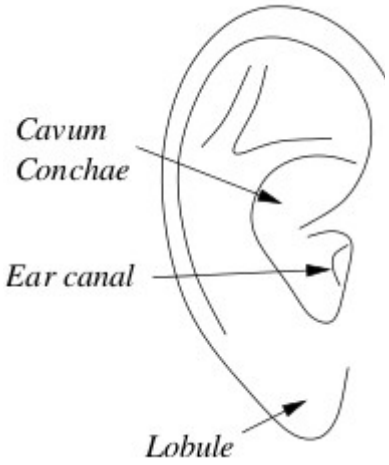

Figura 3: *Orecchio esterno: pinna*

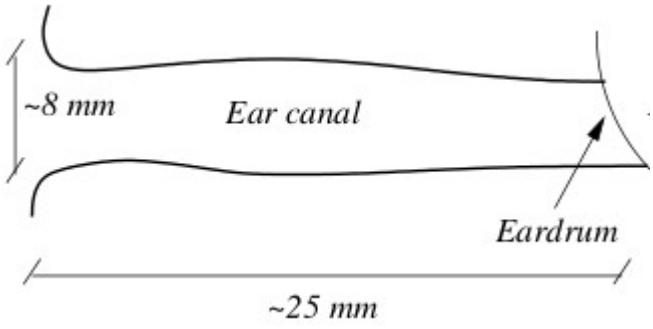

Figura 4: *Orecchio esterno: condotto uditivo*

Differenze di intensità nei due timpani sono più marcati per suoni ad alta frequenza, cioè con una piccola lunghezza d'onda. Tali suoni vengono riflessi dal torso e oscurati dalla testa. L'intensità nelle due orecchie dipende da come è posizionata la testa(banalmente l'orecchio diretto verso la sorgente ha un'intensità maggiore). Il tempo di interarrivo alle due orecchie dipende dalla frequenza però, ovviamente, il suono arriva prima all'orecchio più vicino alla sorgente.

Questo effetto di filtraggio della pinna può essere interpretato studiando le alterazioni introdotte oppure concentrandosi nel dominio della frequenza. Le modifiche sonore introdotte dalla pinna sono:

- riflessioni: in Figura 5a sono illustrate due differenti direzioni di arrivo, entrambi i casi vi sono un percorso diretto ed uno pi` lungo (dovuto appunto alla riflessione della pinna) dalla sorgente al canale uditivo. A frequenze moderatamente basse la pinna convoglia energia sonora e i segnali dei due per-

corsi arrivano in fase; a frequenze alte il ritardo tra i due segnali introduce uno sfasamento che provoca un'interferenza distruttiva. La maggiore interferenza si ottiene quando la differenza di lunghezza dei due percorsi equivale a $\lambda/2 + k\lambda$ (con $k \in \mathbb{N}$ e $k > 0$), cioè quando i due segnali sono in opposizione di fase: questo produce un "pinna notch".

- risonanze: lo studio di queste (Figura 5b) si può effettuare attraverso una più accurata analisi delle misurazioni di risposte in frequenza, eseguite utilizzando un'imitazione della pinna e del canale uditivo con una terminazione in alta impedenza. Le risonanze sono dovute sia alla *cavum conchae* che al prolungamento del canale, circa del 33%, dovuto alla pinna.

Supponiamo che la sorgente del suono sia dritto davanti a noi ma variabile in altezza. In questo caso, ci potrebbero non essere differenze di intensità alle due orecchie. Ciononostante, un ascoltatore riesce comunque ad individuarne la provenienza. Batteau(1967) mise in chiaro che se l'ascoltatore piega una delle sue orecchie perde completamente la capacità di localizzare la sorgente nel piano mediano[4].

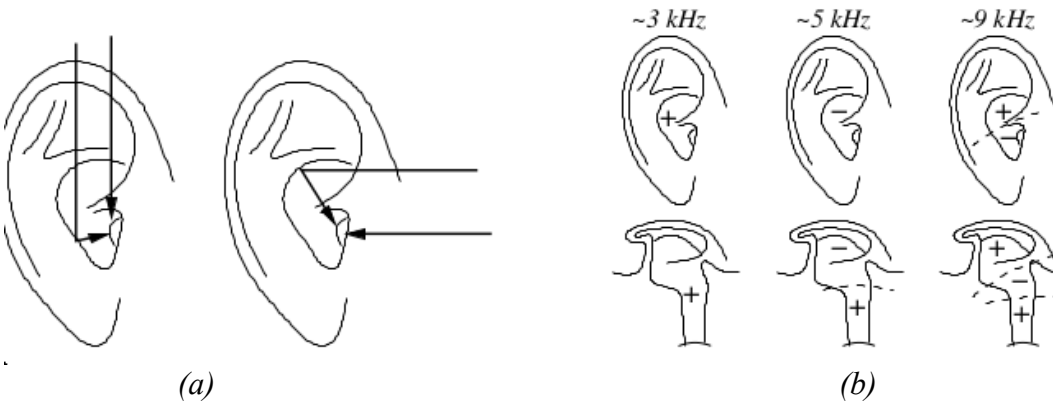

Figura 5: *Effetti della pinna: (a) riflessioni dipendenti dalla direzione (b) risonanze*

In conclusione si può dire che la pinna e il canale uditivo formano una sorta di risonatore acustico, le cui frequenze di risonanza dipendono dalla direzione e dalla distanza della sorgente.

1.3 Rendering Spaziale

Un sistema stereo a due canali, o anche a quattro canali, non riesce a riprodurre perfettamente il suono che uno o più sorgenti generano in uno ambiente chiuso. Nel sistema stereo, per esempio, variando la potenza del suono da altoparlante all'altro, si riesce a creare l'impressione che la sorgente sonora si muova continuamente tra le due casse. Tuttavia, il suono percepito con questa tecnica non uscirà ma dalla linea tra le due casse (per esempio non si riuscirà a metterlo dietro l'ascoltatore).
Lo scopo di questa tesi è quella di riuscire, con l'uso delle sole cuffie, a riprodurre quel senso di immersione che si prova quando si ascolta, per esempio, della musica in una stanza o si assiste ad un concerto.
Una domanda sorge allora spontanea. Si riesce davvero a riprodurre la stessa esatta pressione acustica prodotta dalle sorgenti che si presenterebbe alle orecchie dell'ascoltatore se questi fosse in una sola concerti o, in altre parole, si riesce a virtualizzare (rendere tridimensionale) il suono con l'utilizzo delle sole cuffie?
Questo problema è stato affrontato nello studio di Schroeder, Gottlob e Siebrasse (1974) costruendo una testa di manichino dotato di l'orecchio esterno (pinna e canale uditivo) e di microfoni posizionati alla fine del canale uditivo[4]. Collegando l'uscita dei microfoni alle cuffie di un ascoltatore umano (uscita del microfono sinistro collegato alla cuffia sinistra e quello destro alla cuffia destra) si riesce ad approssimare i segnali nei canali uditivi dell'ascoltatore come se fosse effettivamente presente nel luogo dove è stato effettuata la registrazione.
Questo approccio non è adatto a tutti. Il motivo principale è che l'orecchio esterno varia da individuo a individuo, quindi l'effetto sarebbe migliore se si riuscisse ad effettuare la registrazione direttamente con la propria testa piuttosto che quella del manichino. Due segnali diversi vengono consegnati alle due orecchie; questo è un esempio di segnale *binaurale*;
Con rendering spaziale del suono si intende quindi l'abilità di trasmettere all'ascolatatore e riprodurre con sistema audio (ad esempio cuffie o surround) quella sensazione di immersione che un ascoltatore avrebbe se fosse effettivamente in presenza della

sorgente del suono. Si parla quindi di virtualizzazione del suono.

Come già discusso in precedenza, virtualizzare un suono significa riuscire a riprodurre le stesse condizioni (la stessa pressione acustica), stessi segnali, ai due timpani, che produrrebbe un suono reale posizionato in un particolare punto nello spazio.

La sorgente è spesso individuata dalle coordinate sferiche: azimuth θ, elevazione φ e distanza r. La Figura 6 (Coordinate sferiche usate nella definizione della HRTF: vertical-polar coordinate system) illustra questo sistema di riferimento nel piano verticale.

Ci sono diverse tecniche di rendering spaziale. La più nota è senza dubbio il sistema surround che, con l'utilizzo di diverse sorgenti opportunamente posizionate, riesce a ricreare il senso di immersione nel suono. Questo modo di procedere, se pur non perfetto, ha un grado di errore accettabile, tanto che è la tecnica di gran lunga più diffusa. I problemi più diffusi sono la difficoltà nel posizionamento delle sorgenti e rendere le orecchie mutuamente esclusivi (ovvero eliminare il cross-talk).

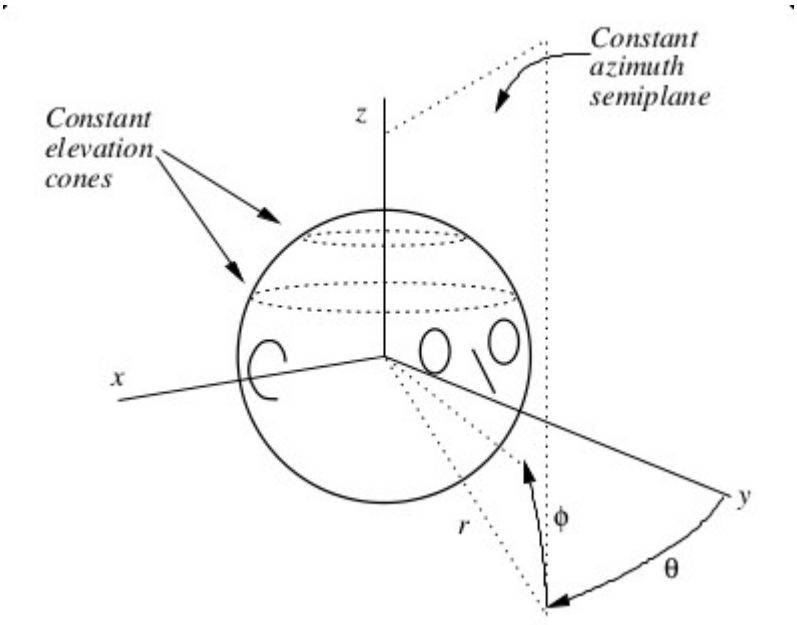

Figura 6: *Coordinate sferiche usate nella definizione della HRTF: vertical-polar coordinate system*

La seconda tecnica, che è quella che d'ora in poi andremo ad analizzare, prevede l'u-

tilizzo degli auricolari. È una tecnica piuttosto complicata perché la forma della stanza, il materiale delle pareti, il corpo dell'ascoltatore (tersa, torso, pinna ecc.) modificano in modo significativo la percezione del suono. Tutte queste differenze devono essere modellati opportunamente nella creazione del segnale binaurale. In questo caso si parla di modello strutturale.

L'utilizzo delle cuffie presenta una serie di svantaggi rispetto ai sistemi ad altoparlanti: sono invasive e scomode da indossare per lunghi periodi di tempo, non hanno una risposta in frequenza piatta che può compromettere la spazializzazione e rendono infine l'idea di sorgenti sonore molto "chiuse" e non compensano il movimento dell'ascoltatore se non con l'utilizzo di un sistema di motion-tracking. Vi sono però due importanti vantaggi: innanzitutto eliminano i riverberi e, in seguito, semplificano di molto le tecniche di 3D sound rendering. Al contrario dei sistemi ad altoparlanti, le auricolari non soffrono del fenomeno di "cross-talk": il suono emesso giungerà sempre ad entrambe le orecchie dell'ascoltatore. Se si ignora l'effetto prodotto dall'ambiente di ascolto si possono utilizzare, in prima approssimazione, tecniche di rimozione del "cross-talk" per adattare i segnali per due altoparlanti stereo alle cuffie.

Nel caso di un sistema binaurale, e quindi basato sull'utilizzo di cuffie, si possono presentare due tipi di localizzazione:

- Inside-the-head localization (IHL, o lateralizzazione): questo termine è tipicamente utilizzato per indicare quando la sorgente sonora è percepita all'interno della testa, e principalmente lungo l'asse interaurale (vedi asse x in Figura 6); si può indurre questa percezione manipolando opportunamente ITD ed ILD in un sistema binaurale. Questo fenomeno illustra un esempio fondamentale di posizionamento virtuale della sorgente nello spazio; infatti se vengono trasmessi due segnali monoaurali perfettamente identici la sensazione sarà che la sorgente sia collocata al centro della testa. Variando ITD ed ILD, a inoltre, si avrà la sensazione che la sorgente si sposti lungo l'asse interaurale da un'orecchio all'altro sempre all'interno della testa.
- Esternalizzazione: è un tipo di localizzazione molto difficile da riprodurre in

un sistema binaurale basato su cuffie, dove risulta molto più semplice l'IHL; infatti non è del tutto chiaro quali siano i parametri supplementari più efficaci per rendere possibile questo tipo di localizzazione, tuttavia si è notato come l'aggiunta di un riverbero, naturale o artificiale, e possa migliorarla sensibilmente.

1.4 HEAD-RELATED TRANSFER FUNCTION (HRTF)

Le HRTFs catturano le trasformazioni subite da un'onda sonora nel suo tragitto verso i timpani, dovute tipicamente alla diffrazione e riflessioni sul torso, pinna, spalle e testa dell'ascoltatore. Una conoscenza profonda dei parametri di processo permette il posizionamento virtuale delle sorgenti sonoro nello spazio: coerentemente con la posizione relativa della testa dell'ascolatatore, il segnale viene filtrato con il corrispondente paio di HRTFs creando cosi' i segnali binauarali da "consegnare" agli auricolari. In questo modo, un ambiente tridimensionale con un alto senso di immersione può essere simulato e integrato in ambienti di *mixed reality.*

L'idea di base dietro al rendering 3D tramite le HRTF è quella di usare HRIR (Head-Related Impuse Response) e HRTF misurate. Dato un segnale anecoico e una posizione virtuale indivuata da (θ e ϕ), un segnale destro ed uno sinistro vengono sintetizzati tramite le due seguenti operazioni

i. ritardando il segnale anecoico di una quantità prefissata si riesce a introdurre l'ITD desiderato.
ii. Facendo la convoluzione del segnale anecoico con le due HRIR (sinistro e destro).

Uno schema sintetico è dato in Figura 7.

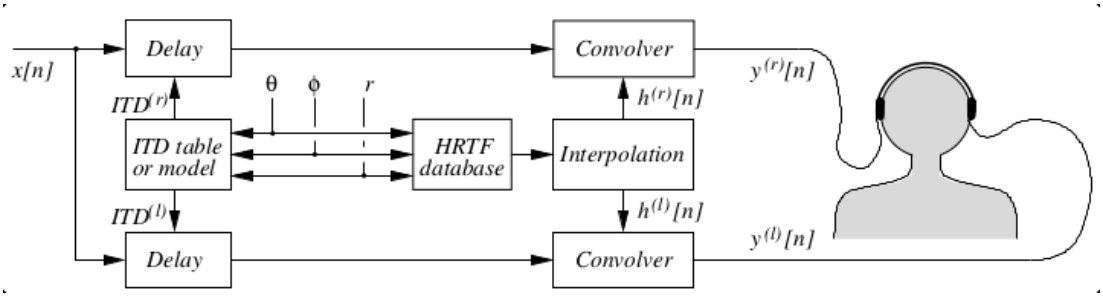

Figura 7: *Schema a blocchi di un sistema di rendering spaziale del suono basato sulle HRTF*

1.4.1 MODELLO STRUTTURALE

Al contrario del metodo di valutazione delle HRTF discusso poco sopra, e di altre tecniche simili, il modello strutturale presentato in questo paragrafo si basa sull'astrazione degli effetti delle diverse parti del corpo che, una volta combinati, danno origine alla HRTF.

Quest'ultima viene poi modellata come una combinazione di blocchi (filtri), ciascuno per il contributo di una parte anatomica. I parametri di ciascun filtro possono essere legati a misure antropometriche (es. la distanza interaurale, o il diametro della cavum conchae), con il vantaggio che una generico modello strutturale per la HRTF può essere adattata ad un specifico ascoltatore e può prendere in considerazione anche la variazione di posizione dell'ascoltatore. Un altro vantaggio è che anche gli effetti dovuti alla stanza possono essere considerati nel modello, in particolare le riflessioni precoci possono essere elaborati tramite il modella della pinna.

Risulta chiaro che separare i contributi delle varie strutture anatomiche in blocchi perfettamente indipendenti è una approssimazione euristica che non tiene in considerazione le varie interazione dovute alle onde riflesse da una struttura all'altra. Tuttavia, le ricerche mostrano che il modello strutturale è in grado di fornire una buona approssimazione di HRTF reali.

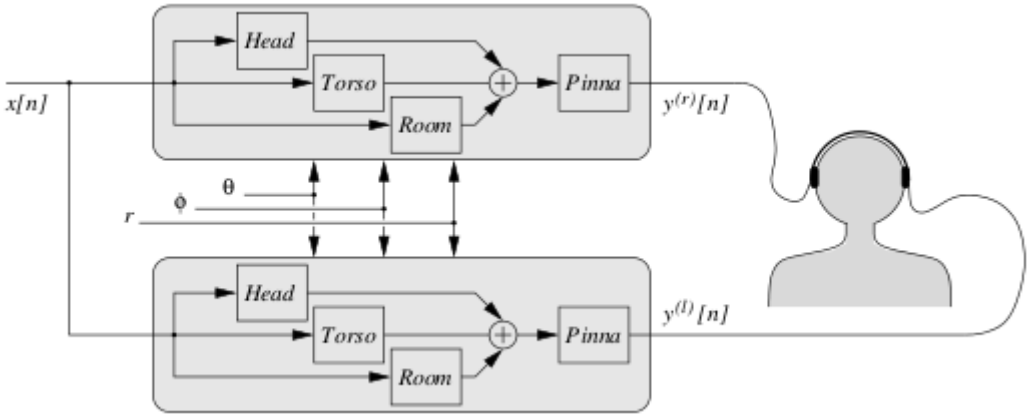

Figura 8: *Schema a blocchi di un sistema di rendering spaziale del suono basato sul modello strutturale*

La Figura 8 mostra un generico modello strutturale. Nel resto di questo paragrafo verrà descritto il blocco relativo alla pinna. Per l'approfondimento sugli altri blocchi si rimanda a [2].

1.4.2 MODELLO PER LA PINNA

Da studi condotti sulla testa [2], si può dedurre che gli effetti più rilevanti della pinna che devono essere presi in considerazione sono quelli dovuti alle riflessioni. Questo significa che la pinna è modellata come una sequenza di filtri FIR, in cui ciascun filtro determina una traccia nello spettro.

Dobbiamo renderci conto che le riflessioni sono dovute a piccole lunghezze d'onda ovvero alle alte frequenze e quindi modellare la pinna tenendo in contro solo le riflessioni è solamente una prima approssimazione.

Per realizzare un modello per la pinna, il tutto si riduce a stimare i ritardi dovuti alle riflessioni e alla sua dipendenza da θ e ϕ, tramite misurazioni di HRIRs/HRTFs, o tramite simulazioni numeriche.

2 PUREDATA

2.1 PANORAMICA

Pure Data è un ambiente di programmazione grafica in tempo reale per processare audio e video. Pure Data è prima di tutto un linguaggio di programmazione perché consente di realizzare algoritmi più o meno complessi come tutti gli altri linguaggi. L'interfaccia con cui il musicista-programmatore parla con PD è grafica, quindi non c'è la necessità di scrivere il codice in un editor di testo, ma si realizzano delle patch combinando fra loro vari tipi di oggetti grafici. Nel gergo dei sintetizzatori analogici una patch rappresentava l'insieme dei collegamenti fra i suoi moduli. PD mutua questo concetto: attraverso una patch si definisce graficamente l'ordine con cui i vari oggetti sono collegati fra loro. Gli algoritmi vengono creati selezionando una serie di entità grafiche all'interno di una finestra, detta patch window (vedi Figura 9).

PD funziona in tempo reale, quindi gli algoritmi sono interattivi e i parametri possono essere modificati durante l'esecuzione. E' anche possibile cambiare la struttura stessa di tali algoritmi mentre sono attivi, aggiungendo o rimuovendo moduli in modo semplice e intuitivo. Nasce con la funzione di creare applicazioni audio e, da qualche tempo, video. Una foltissima comunità di sviluppatori, musicisti, hackers e appassionati sforna ogni giorno nuove applicazioni e librerie che potenziano e aumentano le funzionalità di questo ambiente versatile e libero. Infatti Pure Data è un software libero, i suoi sorgenti possono essere scaricati, studiati, modificati e redistribuiti da chiunque. Inoltre Pure Data è multipiattaforma, quindi gira sui sistemi operativi più comuni: GNU/Linux, Microsoft Windows, Apple Mac OS X, FreeBSD.

PD è stato scritto nel 1996 da Miller Puckette, lo stesso che a metà degli anni '80 aveva sviluppato Max, in seguito divenuto un software commerciale. PD riprende le idee e i concetti di quest'ultimo, pur basandosi su una filosofia più libera e aperta che ne fa un sistema più dinamico e dalle prospettive future più interessanti.

2.1.1 PD-EXTENDED

Allo stato attuale PD è arrivato alla versione standard 0.42.5, detta Vanilla, scaricabile dal sito del suo creatore. Il consiglio di chi scrive è però quello di scaricare la versione extended, che contiene non solo il programma standard, ma anche numerose librerie aggiuntive che estendono notevolmente le sue funzionalità. Attualmente *PD-extended* è alla versione 0.41.45 .

2.2 PD WINDOW E PATCH WINDOW

All'apertura di PD compare la finestra principale del programma (PD window: Figura 9) che ha due funzioni principali:

i. mostrare dei messaggi
ii. consentire la configurazione dell'audio e del MIDI e definire i percorsi in cui PD cerca le librerie da caricare

Alcuni messaggi vengono visualizzati in fase di avvio del programma, riguardano le librerie esterne caricate e gli eventuali errori nella configurazione dell'audio o del MIDI. Altri messaggi possono essere mostrati nella PD window durante la normale esecuzione del programmano e il più delle volte riguardano errori di procedura o comunicazioni prodotte dall'utente tramite l'oggetto print che sarà esaminato successivamente. La configurazione dell'audio avviene tramite la voce di menù Media che consente di impostare il MIDI (MIDI settings), l'audio (Audio settings), di avviare il *motore DSP* (Audio on/off) e di testarne il funzionamento (test Audio and MIDI). Generalmente all'apertura del programma ci si trova di fronte soltanto la PD window. Per iniziare a scrivere gli algoritmi è necessario aprire una finestra di patch dal menù File/new (vedi Figura 10).

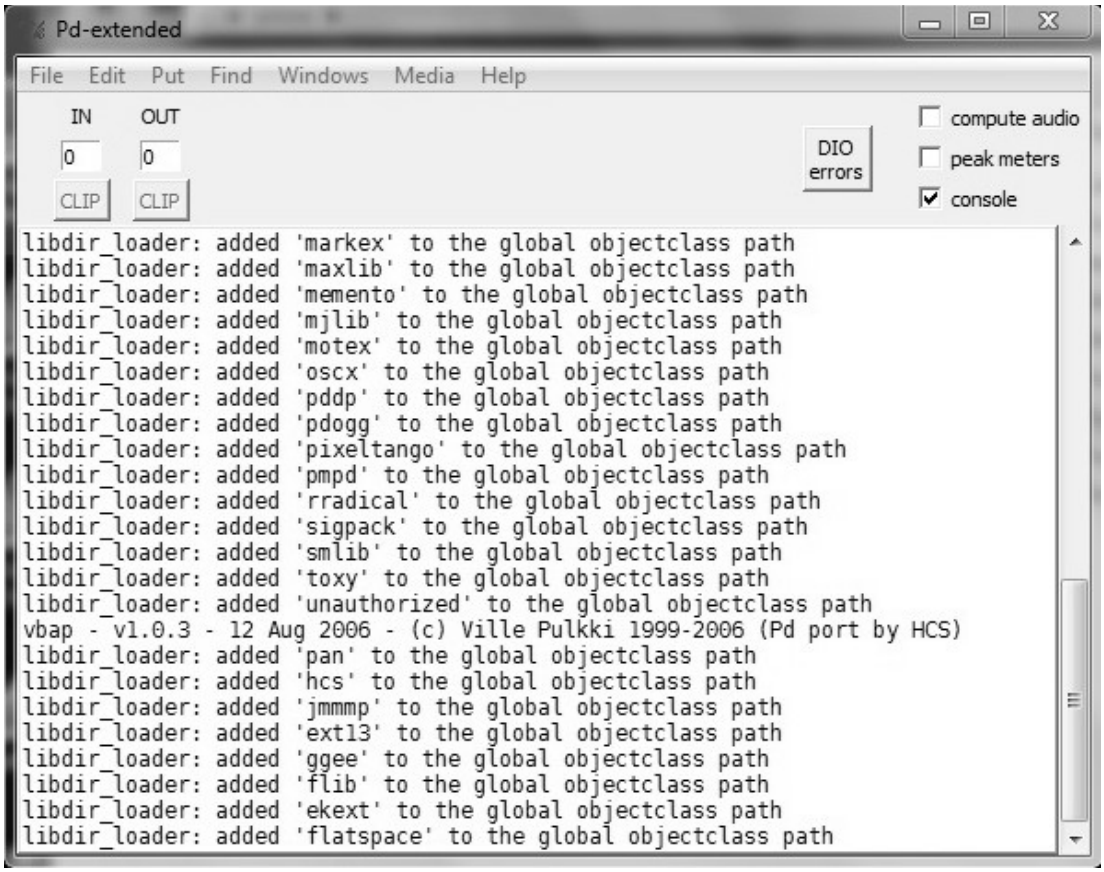

Figura 9: *PD window: finestra mostrata all'apertura di PD*

Compare così la patch window che è l'ambiente di programmazione vero e proprio. Il suo menù è molto simile a quello della PD window ma ovviamente più orientato all'editing delle patch. La patch window può trovarsi in due stati funzionali diversi: *edit mode* e *run mode*. Il primo permette di inserire tutti gli elementi all'interno della finestra, mentre il secondo è necessario per gestire la patch quando questa è in azione. Tutti gli oggetti interattivi cioè quelli che contengono parametri modificabili via mouse o tastiera, funzionano soltanto in *run mode*, mentre in *edit mode* possono solo essere aggiunti o rimossi. Per passare da uno stato all'altro si usa la combinazione di tasti *ctrl-E.*

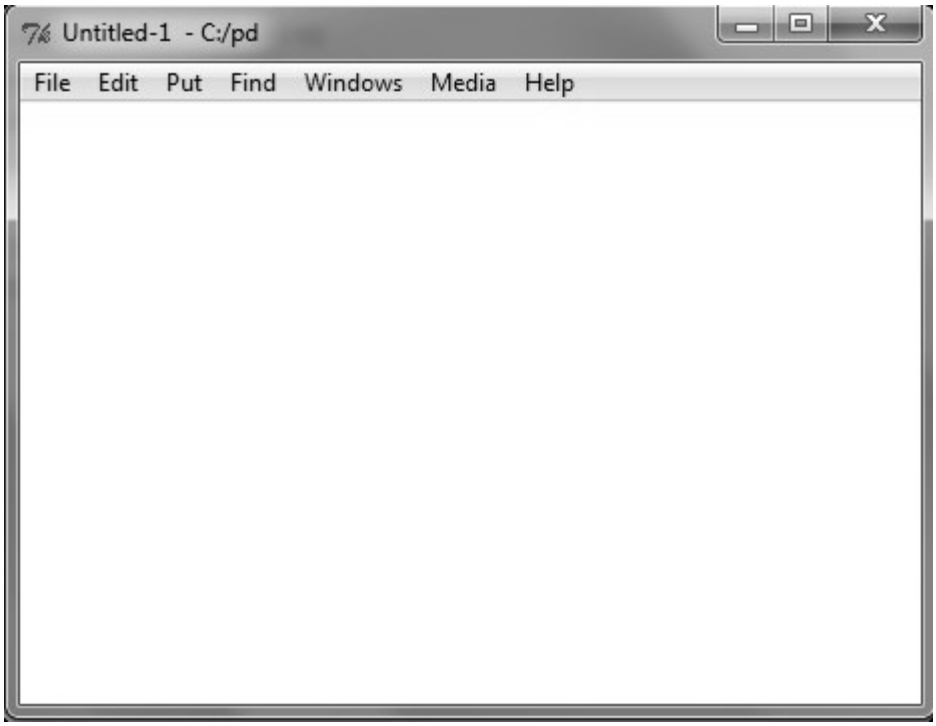

Figura 10: patch window

2.3 Il motore DSP

Quando gli algoritmi di PD processano esclusivamente dati, il programma è completamente attivo sin dalla sua apertura. Nel caso in cui invece si devono processare segnali audio, è necessario attivare il motore DSP per ascoltare il risultato delle operazioni sui segnali. Il motore DSP, acronimo di Digital Signal Processor, si occupa di elaborare il segnale in tempi rapidissimi e di permettere la sua trasformazione da digitale ad analogico e viceversa. Nel momento in cui lo sviluppatore vuole far suonare una patch deve quindi attivare il motore DSP, mediante l'apposita voce di menu Media/audio on. In alternativa può premere la combinazione di tasti ctrl-/.

2.4 Le scatole di PD

La finestra di patch è il luogo che permette la realizzazione degli algoritmi di Pd. Essendo un ambiente grafico la finestra si riempirà di entità di varia natura, dette

scatole (box). Queste scatole sono di quattro tipi: oggetti, messaggi, GUI e commenti e si creano dal menù Put oppure premendo ctrl+n dove n è 1 per gli oggetti, 2 per i messaggi, 5 per i commenti. I numeri 3 e 4 creano delle GUI particolari, simboli e number box che esamineremo successivamente.

2.4.1 OGGETTI E CONNESSIONI

Gli elementi fondamentali della programmazione in Pure Data sono gli oggetti, rappresentati dalle object box, caratterizzate dalla forma rettangolare e dalla presenza di entrate (inlets), nella parte superiore, e di uscite (outlets), nella parte inferiore.
Un oggetto può creare o processare dati oppure segnale audio e riceve attraverso gli inlets messaggi, liste o uscite di altri oggetti. Può inviare dati o segnali ad altri oggetti.

2.4.2 PRINT

Durante l'esecuzione delle patch può accadere che vengano automaticamente visualizzati dei messaggi nella PD window, in particolare alla presenza di errori, ma c'è un oggetto che permette al programmatore di visualizzare nella PD window il messaggio o dato che desidera. Si tratta di print, che può solo ricevere dati o messaggi e stamparli a video nella PD window. In alcune circostanze è utile per verificare il corretto funzionamento degli algoritmi.

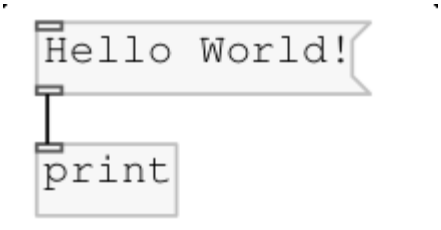

Figura 11: *un click del mouse sul messaggio produce un output nella PD window*

2.4.3 Oggetti più comuni

Verranno qui presentati in tabella alcuni oggetti, che verranno utilizzati in seguito, con il loro simbolo e la rispettiva funzione.

bang	Quando cliccato invia un impulso che attiva gli oggetti ai quali è collegato
loadbang	Esattamente come bang tranne per il fatto che non è necessario cliccarci in quanto viene attivato all'apertura della patch
+~	Somma fra due segnali audio
adc~	Convertitore da segnale analogico a digitale
dac~	Convertitore da segnale digitale ad analogico
print	Mostra l'output nella PD window(console)
vslider	Permette di scorrere un intervallo con “continuità”
osc~	Oscillatore sinusoidale
noise~	Generatore di rumore gaussiano

2.4.4 Esempio di patch PD

Viene qui presentato l'esempio di una patch PD (Figura 12: Esempio di patch PD pe la riproduzione di un file audio in formato wav), che permette di selezionare un file audio in formato *wav* e di riprodurlo su entrambe le cuffie(se stiamo ascoltando con le cuffie). Già qui si vedono le potenzialità di PD: infatti, prima di mandare l'uscita dell'oggetto *readsf~* all'oggetto *dac~*, e quindi alle cuffie, si potrebbe, per esempio, moltiplicarlo per una costante aumentando (o diminuendo se la costante è minore di 1) in questo modo l'intensità del suono. Si potrebbe applicare questa modifica ad una sola degli ingressi di *dac~* creando perciò una differenza di intensità del suono alle due orecchie.

Figura 12: *Esempio di patch PD pe la riproduzione di un file audio in formato wav*

2.5 SEGNALI AUDIO

In Pure Data i segnali audio vengono rappresentati da numeri a 32-bit in virgola mobile, tuttavia sono solitamente limitati a 16 o 24 bit a seconda dell'hardware utilizzato. Sia i valori di input che quelli di output assumono valori compresi fra -1 e 1 e la frequenza di campionamento di default e impostata a 44.100 Hz. Pure Data può leggere o scrivere file audio a 16 o 24 bit a virgola fissa oppure a 32 bit in virgola mobile, nei formati WAV, AIFF o AU tramite gli oggetti soundfiler, readsf~ e writesf~. L'elaborazione dei segnali audio viene eseguita dagli oggetti che riportano nel nome il simbolo ~; questi oggetti comunicano tra loro tramite connessioni audio. Gli inlet e gli outlet sono in generale predisposti sia per segnali audio che per dati di controllo, ma dipende dalle specifiche dell'oggetto. Non e possibile connettere un outlet audio ad un inlet che non possa ricevere audio, e viceversa. Il percorso complessivo del segnale audio in una patch deve essere aciclico; se sono presenti loop, verrà segnalato un errore "DSP loop".

2.5.1 L'OGGETTO DAC~

Negli esempi proposti fino ad ora abbiamo sempre inviato il segnale in uscita all'oggetto dac~ che ha il compito di convertire il segnale digitale in analogico, in modo da poterlo ascoltare. Come si può osservare l'oggetto suddetto ha due inlet. In effetti abbiamo sempre inviato il segnale smistandolo in entrambi gli inlet. dac~ infatti consente di inviare il suono verso due canali, uno sinistro e uno destro. Fino ad ora abbiamo sempre inviato il segnale verso entrambi i canali ottenendo come risultato che il suono sia identico in entrambi, ma potremmo decidere di inviare il segnale verso uno solo dei due.

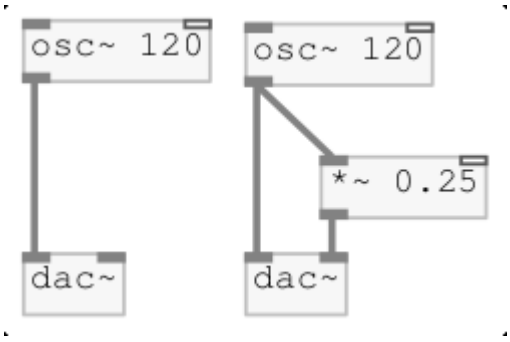

Figura 13: *in (a) il segnale dell'oscillatore è inviato al solo canale sinistro, in (b) il segnale viene inviato con massima ampiezza al canale sinistro, mentre con ampiezza ridotta del 75% al canale destro.*

In Figura 13(a) il segnale viene inviato solo al canale sinistro, mentre in (b) il segnale avrà un peso maggiore sul canale sinistro poiché prima di essere collegato al canale destro viene ridotto in ampiezza.

2.6 INTERNAL, EXTERNAL E LIBRERIE

Per evitare incompresioni di quanto spiegato nel capitolo 3, saranno ora introdotti i significati delle espressioni internal, external e librarie.

- Internal: un *internal* è una classe che è integrato in PD. Molte primitive, come *"+~", "noise~"* o *vslider,* sono degli internal.
- External: un external è una classe non incorporata in PD ma che viene caricata al suo avvio. Una volta caricato in memoria, un external è a tutti gli effetti identico ad un internal
- Libreria: è una collezione di external compilati in un singolo file binario.
  Il nomi delle libreria devono seguire uno specifico formato in base al sistema operativo

libreria	linux	irix	win32
my_lib	my_lib.pd_linux	my_lib.pd_irix	my_lib.dll

  La libreria più semplice contiene esattamente un external con lo stesso nome della libreria.

3 REALIZZAZIONE FILTRI

Come già discusso nel capitolo 1, non c'è alcun dubbio che, se si fissa la direzione della sorgente relativa all'ascoltatore, le differenze più marcate nelle HRTF sono dovute alle caratteristiche della pinna: diverse da individuo a individuo. La pinna ha quindi un ruolo fondamentale nel determinare il contenuto in frequenza delle HRTF grazie a due fenomeni acustici principali,

i. riflessioni lungo i bordi della pinna. Secondo Batteau, le onde sonore vengono tipicamente riflesse dall'orecchio esterno finche la loro lunghezza d'onda è abbastanza piccola rispetto alle dimensioni della pinna stessa, e l'interferenza causata dalle onde provenienti direttamente dalla sorgente e quelle riflesse fanno si' che, nella parte delle alte frequenze, compaiano dei notch acuti nello spettro del segnale ricevuto.

ii. risonanze nelle condotte della pinna. Come discusso da Shaw, poiché la concha si comporta come un risonatore, alcune bande di frequenze, sia delle onde dirette che di quelle riflesse, vengono amplificate in modo significativo in base all'elevazione della sorgente.

Conseguentemente, la parte della HRTF dovuto al contributo della pinna presenta nella sua ampiezza una sequenza di picchi e notchs.
I due external sviluppati per questo progetto hanno la funzione di isolare e valutare separatamente questi due fenomeni.
I risultati mostrano che, mentre la componente risonante è a grandi linee simili da soggetto a soggetto, la componente riflessa è molto dipendente dal soggetto.
Nel modello proposto vengono fatte due ipotesi fondamentali: elevazione e azimuth sono gestite ortogonalmente e i relativi contributi vengono perciò separati in due

parti distinti. Il controllo verticale è associato agli effetti acustici relativi alla pinna mentre il controllo orizzontale è dedicato alla diffrazione della testa.

3.1 COMPONENTE RISONANTE

Le PRTF di qualunque individuo presentano due area di massima risonanza lungo l'asse delle frequenze. La prima, centrata intorno a 4 kHz, sembra essere molto simili fra gli individui poiché si estende per tutte le elevazioni: tuttavia, conoscenze dei modi della pinna comportano che una seconda risonanza probabilmente interferirà in questo intervallo di frequenze. Dal altro lato la seconda area di risonanza, anche se diverso in forma e modulo tra soggetti, è più evidente a basse frequenze comprese tra 12 e 18 kHz.

Poiché le risonanze hanno un comportamento simile tra tutti soggetti analizzati, la personalizzazione di questa parte del modello può essere omessa. Lo spettro dell'ampiezza media (vedi Figura 14) è stato invece calcolato e analizzato per risintesi. Più dettagliatamente, è stata applicata una procedura naive che estrai per tutti gli angoli di elevazione disponibili i due massimi e la media del modulo dallo spettro, dando origine il guadagno G^i_p e alla frequenza centrale CF^i_p di ciascun picco di risonanza, i = 1, 2, e la corrispondente banda a 3dB BW^i_p. Poi, un polinomio del quinto ordine (con l'elevazione ϕ come variabile indipendente) è stato adattato (interpolato) a ciascuno dei tre parametri iniziali dando luogo alle funzioni $G^i_p(\phi)$, $CF^i_p(\phi)$ e $BW^i_p(\phi)$, i = 1, 2. Queste funzioni saranno usate nel modello per il controllo continuo dell'evoluzione della componente risonante quando la sorgente del suono si muove lungo l'elevazione.

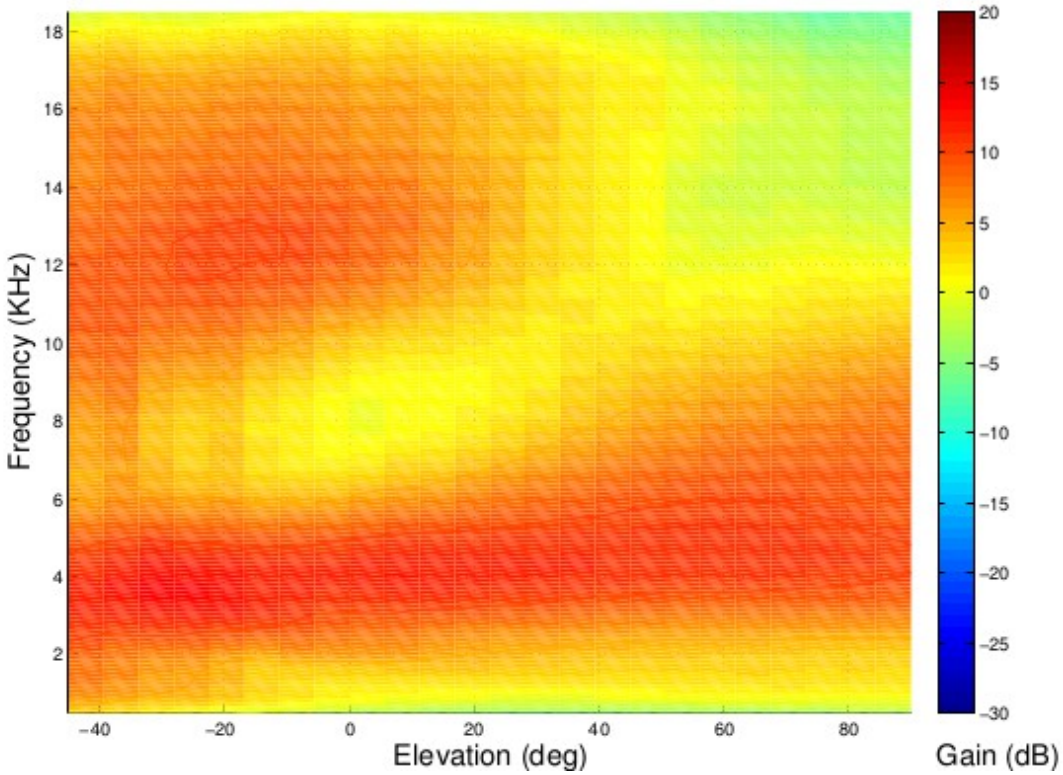

Figura 14: *Spettro dell'ampiezza media della componente risonante della pinna, mediata sulle risposte dell'orecchio sinistro di tutti i 45 soggetti CIPIC.*

3.2 COMPONENTE RIFLESSA

Analisi della parte riflessa mostrano che mentre le PRTF esibiscono una scarsa struttura notch quando la sorgente è sopra la testa, appena l'elevazione decresce, *spectral location,* profondità dei notch crescono fino al punto da differire da individuo a individuo. Ciononostante, tenere traccia delle variazioni dei notch lungo gli angoli di elevazione hanno evidenziato la presenza, nella maggior parte dei soggetti, di tre maggiori (e apparentemente continue) tracce notch tra 5 e 14 kHz, le cui evoluzioni potrebbero essere messi direttamente in relazione con la posizione dei punti di riflessione sulla superficie della pinna. In realtà, assumendo che i coefficiente di tutte le riflessioni che si verificano nella pinna siano negativi, la distanza supplementare percorsa dalle onde riflesse rispetto a quelle dirette deve essere uguale a metà della lun-

ghezza d'onda affinché si verifichi una interferenza distruttiva (cioè un notch), che si traduce in una frequenza di notch che è inversamente proporzionale a tale distanza. Quindi, supponendo che la superficie di riflessione si sempre perpendicolare all'onda sonora, consideriamo la funzione di mappatura

$$d(\phi)=\frac{c}{2\mathrm{CF}_n} \quad , \tag{1}$$

dove d(ϕ) è la distanza dell' ipotetico punto di riflessione dal canale uditivo a elevazione ϕ, CF_n è la frequenza centrale dl notch, e c è la velocità del suono. Mappature dirette delle frequenze di notch lungo le elevazioni tramite la funzione di cui sopra ha mostrato una incoraggiante corrispondenza tra i punti di riflessioni computate e il contorno della pinna vista da un lato della testa.

I risultati ci permettono di eseguire la procedura inversa, abbozzata nelle figure 14, 15 e 16, per riuscire a estrarre le frequenze di notch da una rappresentazione del contorno della pinna. Specificatamente, prima un'immagine è scalata in modo da rispecchiare le dimensioni reali della pinna. Dopodiché i tre contorni più' evidenti e rilevanti vengono tracciati manualmente con l'aiuto di un tablet e memorizzati successivamente come una sequenza di pixel. Queste sono poi tradotte in una coppia di coordinate polari *(d, ϕ),*rispetto al punto dove è stato posizionato il microfono, tramite semplici equazioni trigonometriche. Infine, la frequenza di notch CF_n si deduce facilmente dalla formula precedente semplicemente facendo l'inversa e, similmente al caso delle risonanze, la sequenza (CF^i_n, ϕ) per ciascuna delle tre tracce notch, i = 1,2,3, è approssimata linearmente da un polinomio del quinto ordine $CF^i_n(\phi)$.

Per quanto riguarda gli altri due parametri che definiscono un notch, ovvero il guadagno G_n e la banda a 3dB BW_n, non ci sono ancora prove di corrispondenze con caratteristiche antropometriche. Da un'analisi statistica del primo ordine sulla profondità e banda dei notch si nota una varianza alta tra ciascuna traccia e elevazione, e che la media si mantiene approssimativamente costante tranne per una leggera diminuzione

nella profondità del notch e un aumento di banda quando l'elevazione aumenta (vedi Figura 18).

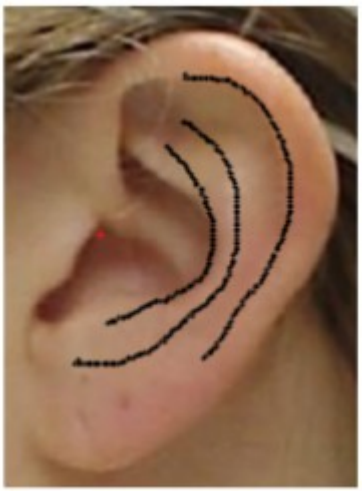

Figura 15: *Estrazione delle frequenze di notch da una foto della pinna (Soggetto 048): (a) Tracciamento dei contorni*

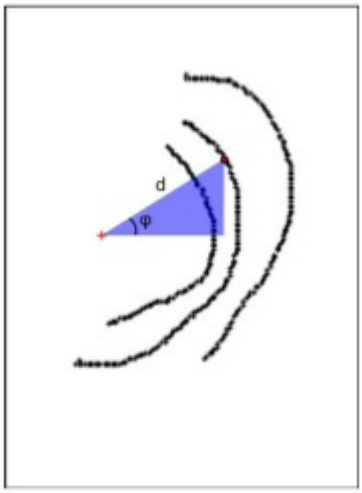

Figura 16: *Estrazione delle frequenze di notch da una foto della pinna (Soggetto 048): Mappatura a coordinate polari*

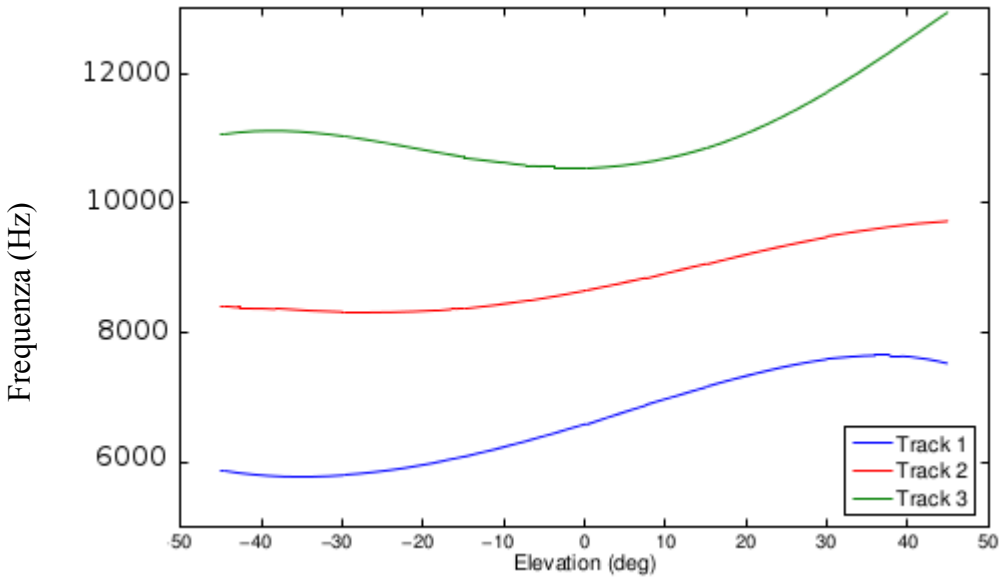

Figura 17: *Estrazione delle frequenze di notch da una foto della pinna (Soggetto 048): Estrazione e approssimazione traccia notch*

In assenza di un chiaro modello dipendente dall'elevazione, viene calcolata la media dei guadagni e bande di tutte le tracce e elevazioni ϕ fra tutti i soggetti e un polinomio del quinto ordine dipendente dall'elevazione viene adattato a ciascuno di questi punti generando le funzioni $G^j_n(\phi)$ e $BW^j_n(\phi)$, $j = 1, 2, 3$.

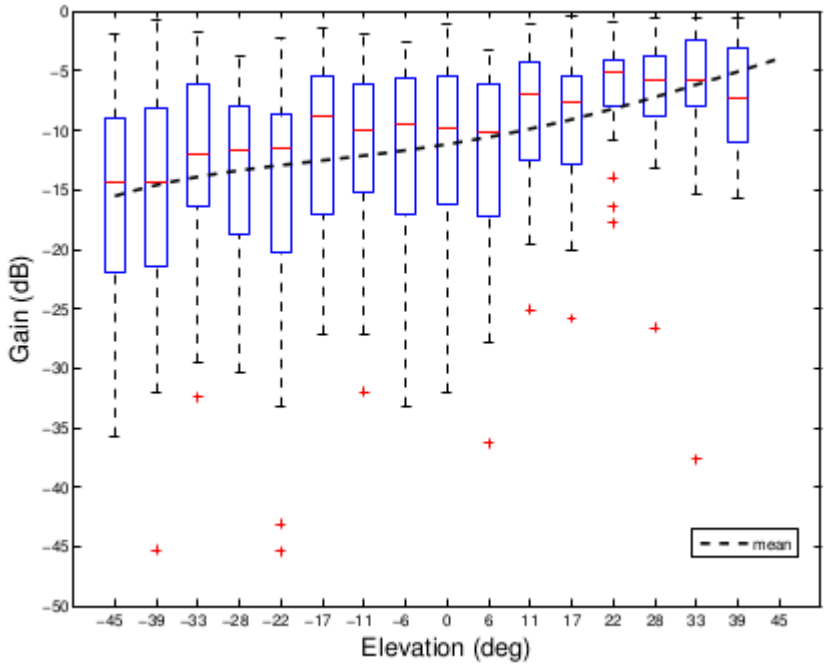

Figura 18: *Box plot e media dei guadagni della prima traccia notch tra i soggetti CIPIC*

3.3 FILTRI

In Figura 19 è visibile l'insieme globale del modello con i relativi parametri ottenuti tramite la procedura di analisi descritta nei paragrafi precedenti. Esaminando la struttura da sinistra verso destra, incontriamo dapprima il modello sferico generico della testa che approssima l'effetto di oscuramente della testa e gli effetti di diffrazione, dove il parametro *a* indica il raggio della testa ottenuto dalla media

pesata su tutti i soggetti CIPIC. Il lavoro presentato in questa tesi si concentra sulla seconda parte, relativa agli effetti della pinna, tenendo presente che esistono diversi lavori di ricerca sulla stima di ITD e ILD che possono essere uniti al modello.

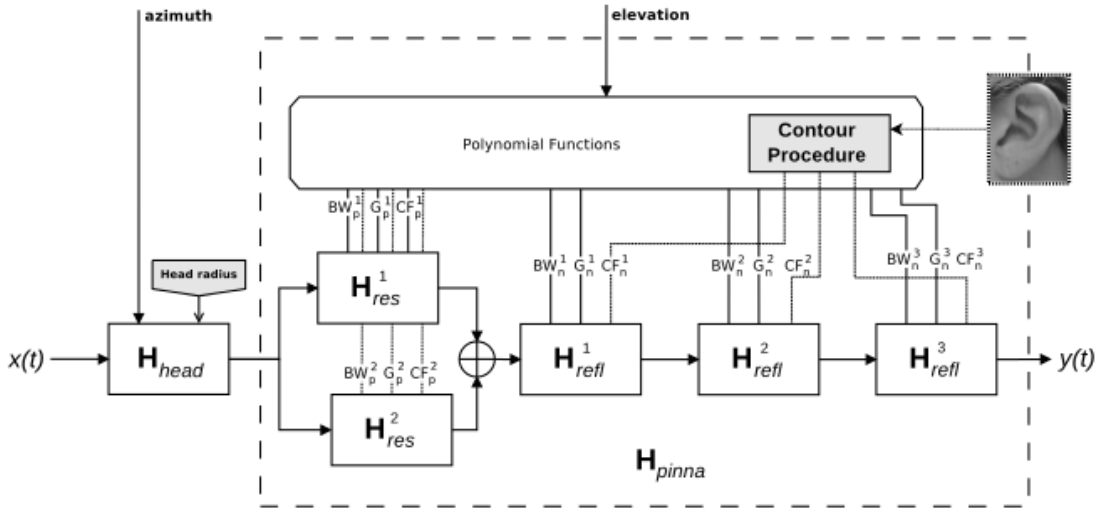

Figura 19: *Modello strutturale per HRTF personalizzate*

L'unico parametro usato dal blocco della pinna è l'elevazione ϕ, che è il parametro indipendente delle funzioni polinomiali che descrivono le frequenze centrali delle risonanze $CF^i_p(\phi)$, le bande a 3 dB $BW^i_p(\phi)$ e i guadagni $G^i_p(\phi)$, i = 1, 2, e i corrispondenti parametri notch ($CF^j_n(\phi)$, $BW^j_n(\phi)$, $G^j_n(\phi)$, j = 1, 2, 3). Solamente la frequenza centrale CF_n è personalizzata in base alle caratteristiche della pinna del soggetto, perciò il polinomio corrispondente deve essere calcolato offline prima di avviare il processo di simulazione.

La parte risonante è modellata dal parallelo di due diversi filtri del secondo ordine per i picchi. Il primo filtro (i = 1) è della forma

$$H^{(1)}_{res}(z) = \frac{1+(1+k)\frac{H_0}{2}+l(1-k)z^{-1}+(-k-(1+k)\frac{H_0}{2})z^{-2}}{1+l(1-k)z^{-1}-kz^{-2}} \quad , \qquad (2)$$

Dove

$$k=\frac{\tan\left(\pi\frac{BW_p^1(\phi)}{f_s}\right)-1}{\tan\left(\pi\frac{BW_p^1(\phi)}{f_s}\right)+1} \quad , \tag{3}$$

$$l=-\cos\left(2\pi\frac{CF_p^1(\phi)}{f_s}\right) \quad , \tag{4}$$

$$V_0=10^{\frac{G_p^1(\phi)}{20}} \quad , \tag{5}$$

$$H_0=V_0-1 \quad , \tag{6}$$

e f_s è la frequenza di campionamento. I secondo picco è nella forma seguente

$$H_{res}^{(2)}(z)=\frac{V_0(1-h)(1-z^{-2})}{1+2\mathrm{lhz}^{-1}+(2\mathrm{h}-1)z^{-2}} \quad , \tag{7}$$

$$h=\frac{1}{1+\tan\left(\pi\frac{BW_p^2(\phi)}{f_s}\right)} \quad , \tag{8}$$

mentre i parametri l e V_0 sono definiti dalle equazione (4) e (5), dove l'indice del polinomio è i = 2. La ragione di questa distinzione si trova proprio nel comportamento a bassa frequenza al quale siamo interessati a modellare. L'implementazione precedente presenta un guadagno unitario alle basse frequenze in modo da preservare tale caratteristica nella struttura parallela del filtro. In questo modo, il filtro integrale della pinna non altera lo spettro del segnale in uscita dal filtro per la testa sferica.

L'implementazione dei filtri del notch è della stessa forma del filtro per il primo picco con la sola differenza nella descrizione dei parametri. Per avere una notazione corretta, i polinomi P_p^l devono essere sostituiti dai polinomi P_n^j, j = 1, 2, 3, e il parametro k definito dall'equazione (3) viene sostituito dalla seguente

$$k = \frac{\tan\left(\pi \frac{BW_n^j(\phi)}{f_s}\right) - V_0}{\tan\left(\pi \frac{BW_n^j(\phi)}{f_s}\right) + V_0} \quad , \tag{9}$$

I tre filtri notch sono in cascata, risultando quindi in un filtro multi-notch di ordine sei.

3.4 IMPLEMENTAZIONE FILTRO IN PD

La Figura 20 (Equivalente PD del blocco Hpinna) è equivalente alla parte relativa alla pinna nel modello matematico in Figura 19 (Modello strutturale per HRTF personalizzate) ovvero a H_{pinna}. Ha come parametri di ingresso, da sinistra verso destra, i seguenti:

- segnale sinistro in uscita da H_{head}
- segnale destro in uscita da H_{head}
- l'elevazione della sorgente
- numero identificativo del soggetto

In base al numero del soggetto, i parametri di guadagno, frequenze centrale e banda vengono caricati da un database precedentemente creato attraverso una procedura di tracciamento dei contorni della pinna del soggetto in questione. Variando quindi l'elevazione in modo lineare, se i valore *CF, G e BW* sono presenti per una data elevazione vengono semplicemente caricati altrimenti si procede con l'interpolazione lineare con il valore immediatamente minore presente in database. Tutte queste operazioni sono pressoché immediate poiché il database viene caricato in memoria e reso sempre disponibile alla creazione dell'oggetto evitando in questo modo rallentameti.
L'uscita della patch *bt_pinna* rappresenta quindi l'uscita *y(t)* ovvero il segnale adattato alle caratteristiche della pinna del soggetto.
Per una prova di simulazione, agli ingressi audio della *bt_pinna* sono stati collegati due *noise~* (un generatore di rumore gaussiano), all'elevazione è stato collegato un *vslider* (vedi 2.4.3 Oggetti più comuni), che ci permette di variare l'elevazione in input alla patch da -45 a +45 gradi, mentre all'ultimo input è stato collegato un oggetto

di tipo *message* che ci permette di indicare il numero del soggetto. L'uscita è collegato all'oggetto *dac~* (vedi 2.5.1 L'oggetto dac~).

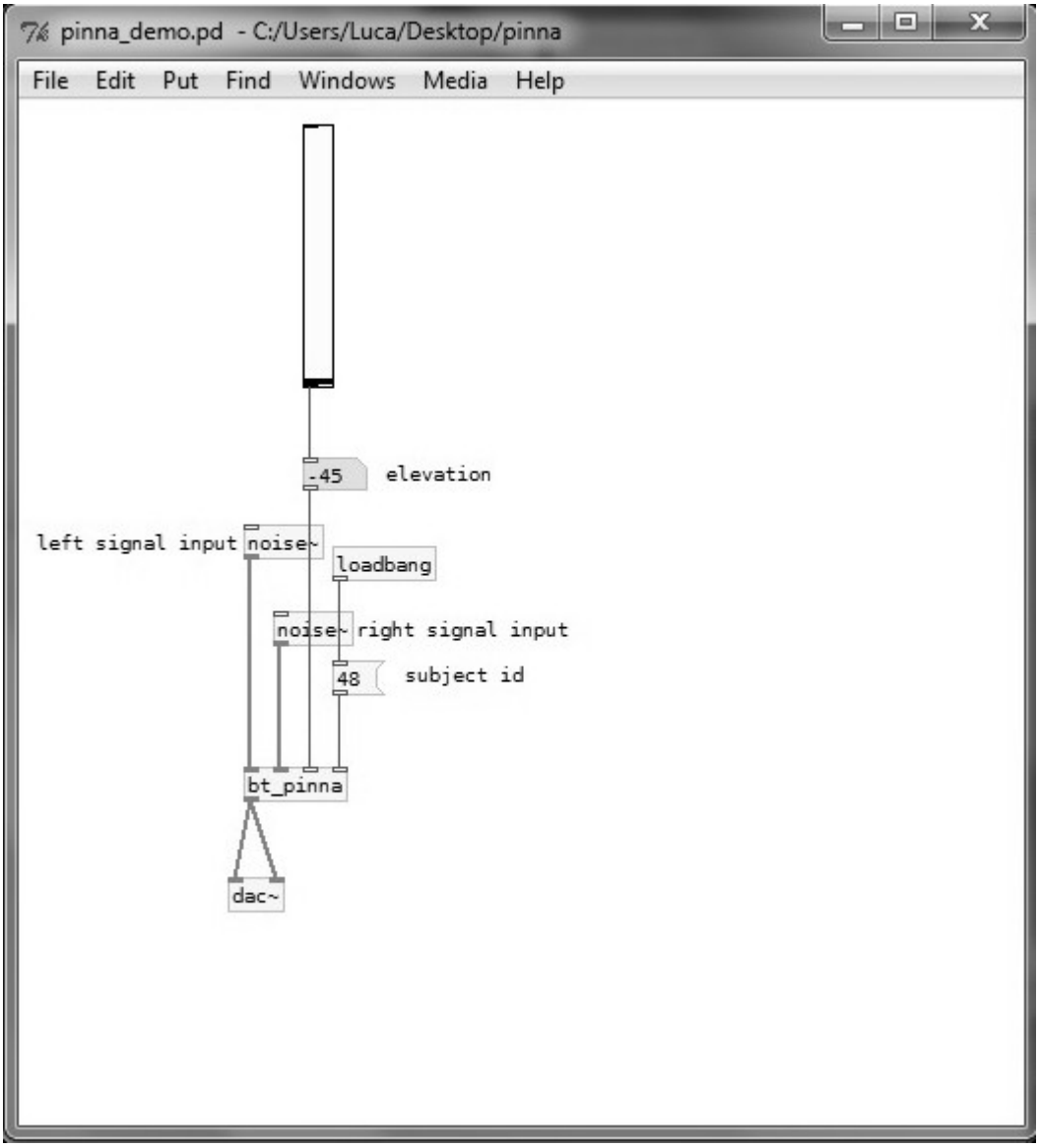

Figura 20: *Equivalente PD del blocco* H_{pinna}

Attivando quindi il *motore DSP* nella PD windows e facendo poi variare l'elevazione in input a bt_pinna tramite il *vslider* si ha l'effetto che la sorgente si stia effettivamente muovendo lungo il piano verticale.

Vediamo ora in dettaglio la patch *bt_pinna (*Figura 21: Patch bt_pinna in dettaglio).

Notiamo subito una somiglianza con il blocco H_{pinna}. Infatti i due external hres~ che modellano le componenti risonanti sono in parallelo e hanno come ingresso i segnali audio indicati dall'oggetto *inlet~*. L'uscita dal parallelo dei due hres~ è quindi usato come input per "*hrefl~ 1"* che modella il primo notch. Gli altri due filtri *hrefl~* sono in cascata.

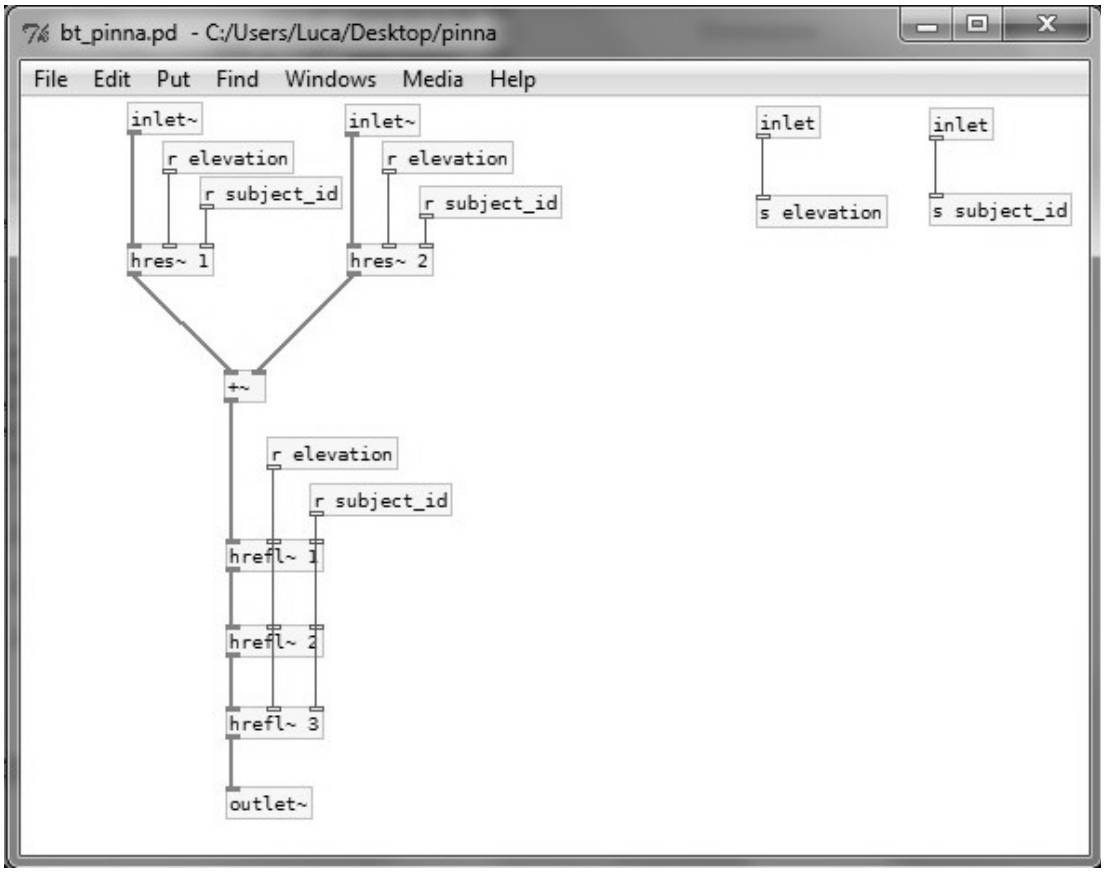

Figura 21: *Patch bt_pinna in dettaglio*

Per l'implementazione di C++ PD dei filtri hres~ e hrefl consultare l'Appendice.

Per la simulazione della patch creata sono stati usati i seguenti dati estratti da simulazioni matlab per il soggetto 048[3].

Prima traccia della componente riflessa:

elevazione(°)	frequenza centrale(Hz)	guadagno(dB)	banda(Hz)
-45	5950	-14.69	245.1
-39.375	5900	-14.35	261.5
-33.75	5880	-13.99	279.2
-28.125	5890	-13.6	298.2
-22.5	5900	-13.18	318.5
-16.875	6000	-12.74	340.1
-11.25	6100	-12.27	363.1
-5.625	6300	-11.77	387.3
0	6500	-11.24	412.8
5.625	6750	-10.69	439.7
11.25	7000	-10.11	467.8
16.875	7150	-9.501	497.3
22.5	7700	-8.866	528
28.125	7800	-8.203	560.1
33.75	7850	-7.513	593.5
39.375	7850	-6.796	628.1
45	7750	-6.052	664.

Seconda traccia componente riflessa:

elevazione(°)	frequenza centrale(Hz)	guadagno(dB)	banda(Hz)
-45	8200	-12.53	357.3
-39.375	8170	-12.96	357
-33.75	8140	-13.32	358

-28.125	8125	-13.6	360.4
-22.5	8125	-13.81	364.2
-16.875	8150	-13.93	369.3
-11.25	8200	-13.98	375.8
-5.625	8350	-13.95	383.7
0	8850	-13.84	392.9
5.625	8900	-13.66	403.4
11.25	9000	-13.4	415.4
16.875	9050	-13.06	428.6
22.5	9125	-12.64	443.3
28.125	9200	-12.14	459.3
33.75	9250	-11.57	476.6
39.375	9500	-10.92	495.3
45	9600	-10.19	515.4

Terza traccia componente riflessa:

elevazione(°)	frequenza centrale(Hz)	guadagno(dB)	banda(Hz)
-45	11045	-16.03	356.2
-39.375	11050	-15.58	363.1
-33.75	11020	-15.13	370.5
-28.125	10995	-14.68	378.3
-22.5	10900	-14.24	386.6
-16.875	10800	-13.79	395.3
-11.25	10700	-13.34	404.5
-5.625	10600	-12.9	414.1
0	10500	-12.46	424.1
5.625	10600	-12.02	434.6

11.25	10800	-11.58	445.6
16.875	10900	-11.14	457
22.5	11150	-10.7	468.8
28.125	11700	-10.26	481.1
33.75	11900	-9.829	493.9
39.375	12300	-9.395	507.1
45	13000	-8.962	520.7

Prima traccia componente risonante:

elevazione(°)	frequenza centrale(Hz)	guadagno(dB)	banda(Hz)
-45	3391	12.5	2733
-39.375	3480	12.46	2636
-33.75	3565	12.42	2554
-28.125	3646	12.37	2486
-22.5	3724	12.31	2432
-16.875	3799	12.26	2392
-11.25	3869	12.2	2367
-5.625	3936	12.13	2356
0	3999	12.06	2359
5.625	4058	11.98	2377
11.25	4114	11.9	2409
16.875	4166	11.82	2456
22.5	4215	11.73	2516
28.125	4259	11.64	2591
33.75	4300	11.54	2681
39.375	4338	11.43	2784
45	4371	11.33	2902

Seconda traccia componente risonante:

elevazione(°)	frequenza centrale(Hz)	guadagno(dB)	banda(Hz)
-45	10830	9.758	3682
-39.375	11180	9.956	3704
-33.75	11530	10.05	3725
-28.125	11870	10.05	3747
-22.5	12190	9.949	3768
-16.875	12510	9.747	3788
-11.25	12820	9.444	3809
-5.625	13120	9.042	3829
0	13410	8.539	3849
5.625	13690	7.936	3868
11.25	13960	7.234	3887
16.875	14220	6.431	3906
22.5	14470	5.528	3924
28.125	14720	4.525	3943
33.75	14950	3.422	3960
39.375	15170	2.219	3978
45	15390	0.9162	3995

3.5 RISULTATI

Il modello proposto è stato testato su diversi soggetti CIPIC; nel seguito presentiamo i risultati per due di essi, soggetto 020(vedi Figura 22) e soggetto 048(vedi Figura 15). in entrambi i casi sono stati scelti gli stessi riferimenti per i contorni.

Figura 22: *Estrazione dei contorni per la pinna del soggetto 020*

Nella simulazione eseguita, tutti i parametri delle funzioni polinomiali forniti al modello sono stati valutati a passo di mezzo grado. Possiamo ora comparare i moduli delle HRTF originali a quelle sintetizzate (vedi figure 23, 24,25 e 26).

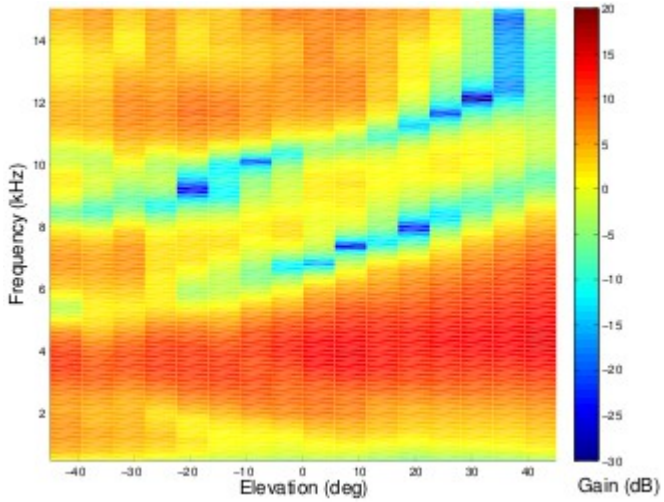

Figura 23: *Grafico originale per il modulo delle HRTF per il soggetto 020*

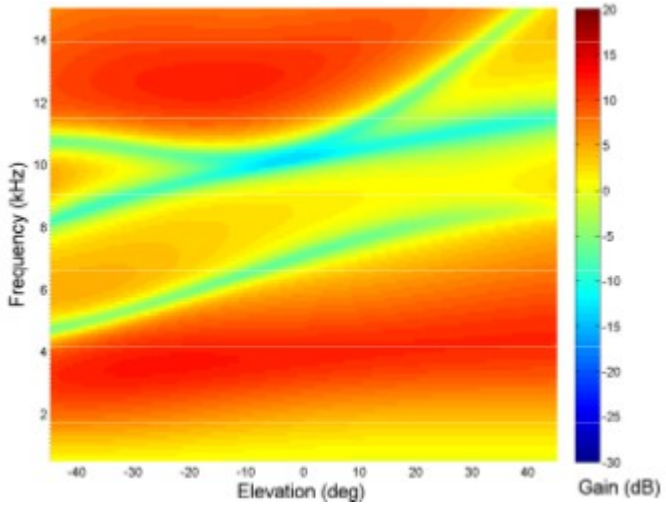

Figura 24: *Grafico sintetico per il modulo delle HRTF per il soggetto 020*

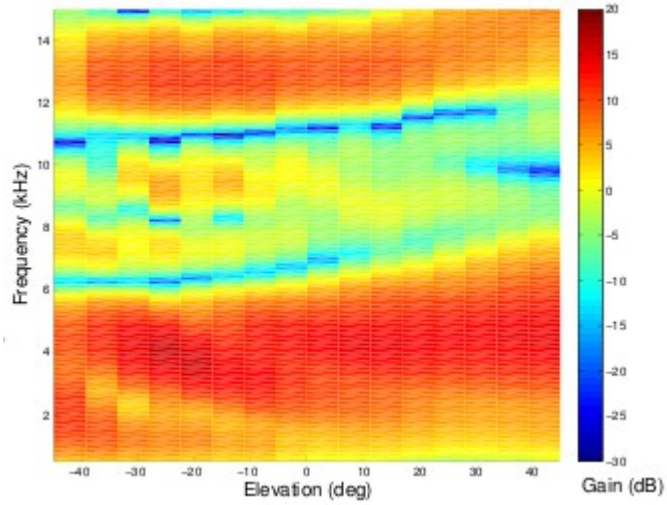

Figura 25: *Grafico originale per il modulo delle HRTF per il soggetto 048*

Ci concentriamo nell'intervallo di frequenze fino a 15 kHz dove tutte le informazioni rilevanti sono presenti, scorrendo tutte le elevazioni comprese tra -45 e 45 gradi nel piano mediano. Oltre a differenze di risoluzione nei grafici originali e in quelli sintetizzati delle HRTF, possiamo osservare caratteristiche simili:

- La prima risonanza, essendo unidirezionale e con un comportamento simile per tutti i soggetti, ha una buona approssimanzione.

- Le tracce notch estratte, anche se molto più regolari rispetto a quelle orginali, seguono fedelmente i valori misurati, dando conferma della correttezza della procedura di estrazione dei contorni e di quella di mapping.
- I guadagni, anche nelle frequenze intermedie tra notch e risonanze, sono gli stessi.

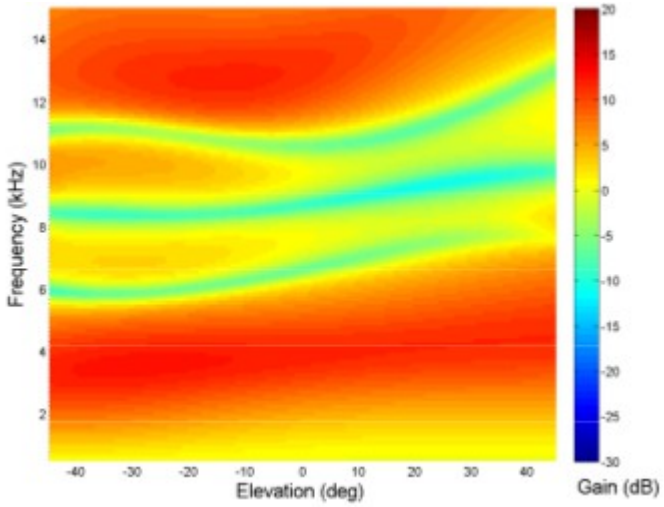

Figura 26: *Grafico sintetico per il modulo delle HRTF per il soggetto 048*

Venendo ora a differenze più sottili, il soggetto 020 presenta un largo notch intorno a 40° nell'intervallo più alto di frequenze che non viene sintetizzato correttamente; questo potrebbe essere dovuta ad una sovrapposizione di due o più notch che non viene scoperta nel processo di estrazione dei contorni.

4 Conclusioni

Il lavoro presentato in questa tesi si inserisce all'interno di un progetto più ampio per il rendering spaziale del suono tramite tecniche binaurali, tendendo in considerazione anche contributi dovuti a caratteristiche antropometriche dell'ascoltatore e alla sua posizione grazie all'utilizzo di sistemi di motion tracking.
È qui presentata la parte relativa alla pinna in un modello strutturale di HRTF personalizzate che può essere usato in un ambiente per il rendering spaziale del suono in tempo reale. Uno dei maggiori vantaggi di un tale approccio, rispetto all'utilizzo di molte risorse per le HRTF misurate, è che il modello può essere parametrizzato secondo informazioni antropometriche dell'utente.
Futuri lavori sul modello per la sintesi della pinna sono orientati verso un miglioramento del controllo verticale tramite l'analisi di una rappresentazione 3D della pinna che permette di esaminare la sua sezione orizzontale. L'equazione semplificata 1 (paragrafo 3.2 Componente riflessa, pagine 33), sul calcolo della distanza delle riflessioni, dovrebbe essere modificata per contenere i contributi dovuti alla pinna anche fuori dal piano parallelo al piano mediano della testa poiché la pinna non è interamente compresa in un piano. Questo miglioramento è cruciale sopratutto per soggetti con le orecchie a sventola[1].
Le aggiunte necessarie per avere una piena ed avvolgente esperienza binaurale spingono la ricerca verso modelli per il posizionamento della sorgente sonora anche dietro, sopra e sotto l'ascoltatore. L'aumento della sensazione di immersione anche del corpo richiede la valutazione dei contributi dovuti alle spalle e al torso, aggiungendo al modello complessivo altre riflessioni ed effetti di oscuramento, specialmente quando la sorgente è sotto l'ascoltatore. Ciononostante, questo modello preliminare può introdurre un reale controllo 3D della sorgente in numerose applicazioni “frontali”.

APPENDICE

A.1 HRES~

A.1.1 FILTRO HRES DI TIPO 1 (*HRES~ 1*)

```
static void hres_tilde_set_filter_parameters_type1(t_hres_tilde *x, int pos){
        int ord;
        float g, vo, h, fc, fb, d;// d0, d1;
        //Calcolo di FC
        int i = 0;
        fc = hres_tilde_interp(pos,i).getFc();
        d = -cos(2*PI*fc / sys_getsr() );

        //Calcolo di Fb
        fb = hres_tilde_interp(pos,i).getFb();
        h= ( 1 ) / ( 1+tan(PI*fb/sys_getsr()) );        //sys_getsr : get sample rate (fs)

        //Calcolo di G gain
        g = hres_tilde_interp(pos,i).getG();
        vo = pow(10,g/20);

        //ordine del filtro
        ord = 3;
        xx1=(float*)malloc(sizeof(float)*ord);
        yy1=(float*)malloc(sizeof(float)*ord);
        for(int i=0;i<ord;i++){
                xx1[i]=0;
                yy1[i]=0;
```

```
        }
        b[0]=vo*(1.0-h);
        b[1]=0.0;
        b[2]= -vo*(1.0-h);
        a[0]=1.0;
        a[1]=2.0*d*h;
        a[2]=2.0*h-1.0;
        hres_tilde_filter(ord, a, b, ord, xx1, yy1);
}
```

A.1.2 FILTRO HRES DI TIPO 2 (*HRES~ 2*)

```
static void hres_tilde_set_filter_parameters_type2(t_hres_tilde *x, int pos){
        int ord;
        float g, vo, fc, fb, d, k, ho;
        //Calcolo di FC
        int i = 0;
        fc = hres_tilde_interp(pos,i).getFc();

        d = -cos(2*PI*fc / sys_getsr() );

        //Calcolo di Fb
        fb = hres_tilde_interp(pos,i).getFb();

        k= ( tan(PI*fb/sys_getsr())-1 ) / ( tan(PI*fb/sys_getsr())+1 );

        //Calcolo di G gain
        g = hres_tilde_interp(pos,i).getG();

        vo = pow(10,g/20);
        ho = vo - 1;

        //ordine del filtro
        ord = 3;
        xx1=(float*)malloc(sizeof(float)*ord);
        yy1=(float*)malloc(sizeof(float)*ord);
        for(int i=0;i<ord;i++){
                xx1[i]=0;
                yy1[i]=0;
        }
```

```
        b[0]=1+(1+k)*ho/2.0;
        b[1]=d*(1-k);
        b[2]=-k - (1+k)*ho/2.0;
        a[0]=1;
        a[1]=d*(1-k);
        a[2]=-k;
        hres_tilde_filter(ord, a, b, ord, xx1, yy1);
}
```

A.1.3 CALCOLO OUTPUT FILTRO HRES~

```
/* hres_tilde_filter: Funzione che si comporta come il filter di matlab
        int ord ordine del filtro
        float *numeratore array che contiene il denominatore del filtro
        float *denominatore array che contiene il numeratore del filtro
        int np lunghezza dell'array in ingresso
        const float *x arraay in ingresso
        float *y array che contiene il risultato del filtraggio */
static void hres_tilde_filter(int ord, float *denominatore, float *numeratore, int np,
const float *x, float *y){
        int i,j;
        y[0] = numeratore[0] * x[0];
        for (i = 1;i < ord + 1; i++){
                y[i] = 0.0;
                for (j = 0;j < i + 1; j++)
                        y[i] = y[i] + numeratore[j] * x[i - j];
                for (j = 0; j < i; j++)
                         y[i] = y[i] - denominatore[j + 1] * y[i - j - 1];
        }

        for (i = ord + 1; i < np + 1; i++){
                y[i] = 0.0;
                for (j = 0; j < ord + 1; j++)
                        y[i] = y[i] + numeratore[j] * x[i - j];
                for (j = 0;j < ord; j++)
                        y[i] = y[i] - denominatore[j + 1] * y[i - j - 1];
        }
}
```

A.1.4 FUNZIONE INTERPOLAZIONE

Questa funzione di interpolazione è la stessa sia per hres~ che per hrefl~. Infatti basta scambiare peak con notch per ottenere la funzione di interpolazione per hrefl~.

```
/*
Effettua l'interpolazione degli elementi della classe PEAK
pos: indice più vicino nella matrice PEAK
i : quale peak bisogna considerare. Sempre 0
*/
static Peak hres_tilde_interp(int pos,int i){
        Peak* out=new Peak();
        if(elevation == ((pos/1)*5.625-45)){
        //non c'e' bisogno di interpolazione valore presente in matrice
        out->setFc(peak[pos+i].getFc());
        out->setG(peak[pos+i].getG());
        out->setFb(peak[pos+i].getFb());
        }else{
        //interpolazione
        if(elevation > ((pos/1)*5.625-45)){
        //elevation e' piu' grande del valore in matrice-> interp con pos+1
        //modifico i fc fb d del filtro
        //solo per la frequenza. Se il successivo valore da interpolare e' 0
        //prendo il valore precedente presente nella matrice dei peak, non effettuo la
        //modifica
        //Mantieni fc pari a 0
        if( peak[pos+i].getFc() == 0 )
        {
                out->setFc(0);
        }else if ( peak[pos+1+i].getFc() == 0 )
```

```
{
        out->setFc(peak[pos+i].getFc());
}else
{
        out->setFc(hres_tilde_linearInt(elevation,((pos+1)/1)*5.625-45,
            ((pos)/1)*5.625-45,peak[pos+1+i].getFc(),peak[pos+i].getFc()));
}
out->setG(hres_tilde_linearInt(elevation,((pos+1)/1)*5.625-45,
    ((pos)/1)*5.625-45,peak[pos+1+i].getG(),peak[pos+i].getG()));
out->setFb(hres_tilde_linearInt(elevation,((pos+1)/1)*5.625-45,
    ((pos)/1)*5.625-45,peak[pos+1+i].getFb(),peak[pos+i].getFb()));

}else{
        //elevation e' piu' piccolo del valore in matrice-> interp con pos-1
        //modifico i fc fb d del filtro
        if( peak[pos+i+1].getFc() == 0 )
        {
                out->setFc(peak[pos+i].getFc());
        }else if ( peak[pos+i].getFc() == 0 )
        {
                out->setFc(0);
        }else
        {
                out->setFc(hres_tilde_linearInt(elevation,((pos)/1)*5.625-45,
                ((pos-1)/1)*5.625-
                45,peak[pos+i].getFc(),peak[pos1+i].getFc()));
        }
        out->setG(hres_tilde_linearInt(elevation,((pos)/1)*5.625-45,((pos-
                1)/1)*5.625-45,peak[pos+i].getG(),peak[pos-1+i].getG()));
        out->setFb(hres_tilde_linearInt(elevation,((pos)/1)*5.625-45,((pos-
```

```
                        1)/1)*5.625-45,peak[pos+i].getFb(),peak[pos-1+i].getFb()));
        }
        }
        return *out;
}
```

A.1.5 CREAZIONE DI HRES~ IN PD

Funzione usata per la creazione degli inlets e outlets (input e output) di *hres~*.

```
static void *hres_tilde_new(t_floatarg f){
        t_hres_tilde *x_x = (t_hres_tilde *)pd_new(hres_tilde_class);
        x_x->default_input = 0;
        x_x->params = t_hres_params();

        //store hres~ type [1 or 2]
        x_x->hres_type = (f > 1.0) ? 2 : 1;
        post((char *)"hres~ type: %d", x_x->hres_type);

        // elevation inlet
        inlet_new(&x_x->x_obj , &x_x->x_obj.ob_pd, &s_float, gensym((char
                *)"elevation_set"));

        // subject id inlet
        inlet_new(&x_x->x_obj , &x_x->x_obj.ob_pd, &s_float, gensym((char
        *)"subject_id_set"));


        //adds new signal outlet
        outlet_new(&x_x->x_obj, &s_signal);


        //caricare struttura dati PEAK
        int i;
        / ord e' il numero di coeff. del filtro.
        // filtro 1 grado -> 2
```

```
        // filtro 2 grado -> 3
        ord=3;
        //Array di 17 notch = (Nntch) 1 * (Nang) 17
        peak=(Peak*)malloc(sizeof(Peak)*Nang*Nntch);
        for(i=0;i<Nang*Nntch;i++){
                peak[i].setFc(0.0);
                peak[i].setG(0.0);
                peak[i].setFb(0.0);
                peak[i].setE(0.0);
        }

        //Array che contengono i valori passati di ingresso x e uscita y.
        x=(float*)malloc(sizeof(float)*7);   //IN
        y=(float*)malloc(sizeof(float)*7);       //OUT
        for(i=0;i<7;i++){
                x[i]=0;
                y[i]=0;
        }

        return (void *)x_x;
}
```

Prima funzione evocata da PD alla creazione dell'oggetto.

```
extern "C" void hres_tilde_setup(void){
        // creation of the hres~ instance
        hres_tilde_class = class_new(gensym((char *)"hres~"),
                (t_newmethod)hres_tilde_new, 0, sizeof(t_hres_tilde),
                CLASS_DEFAULT,   A_DEFFLOAT, 0);
```

```
    // sound processing
    class_addmethod(hres_tilde_class, (t_method)hres_tilde_dsp, gensym((char
            *)"dsp"), (t_atomtype)0);

    // elevation processing
    class_addmethod(hres_tilde_class, (t_method)hres_tilde_elevation_set,
            gensym((char *)"elevation_set"), A_DEFFLOAT, (t_atomtype) 0);

    // subject ID processing
    class_addmethod(hres_tilde_class, (t_method)hres_tilde_subject_id_set,
            gensym((char *)"subject_id_set"), A_DEFFLOAT, (t_atomtype) 0);

    // set help file path
    class_sethelpsymbol(hres_tilde_class, gensym((char
            *)"doc/5.reference/hres~-help"));

    CLASS_MAINSIGNALIN(hres_tilde_class, t_hres_tilde, default_input);
}
```

A.1.6 Oggetto Peak

Classe contenente i parametri di un peak. È identico alla classe per i notch semplicemente scambiando i nomi.

```
class Peak{
	private:
		float fc;
		float g;
		float fb;
		float e;

	public:
		Peak(){
			fc=0.0;
			g=0.0;
			fb=0.0;
			e=0.0;
		}
		 float getFc();
		float getG();
		float getFb();
		float getE();
		void setFc(float c);
		void setG(float gg);
		void setFb(float b);
		void setE(float ee);
};

void Peak::setFc(float c){
	fc=c;
```

```
}
void Peak::setFb(float b){
        fb=b;
}
void Peak::setG(float gg){
        g=gg;
}
void Peak::setE(float ee){
        e=ee;
}
float Peak::getFc(){
        return fc;
}
float Peak::getFb(){
        return fb;
}
float Peak::getG(){
        return g;
}
float Peak::getE(){
        return e;
}
```

A. 2 HREFL~

A.2.1 FILTRO HREFL~

La maggior parte delle funzione di hrefl~ è simile ai metodi già ripostati per *hres~* quindi vengono omessi.

```
/*
Creazione del filtro multi notch
*/
static void href_tilde_createFilter(int pos){
        float k,l,vo,ho;
        float a[3],b[3],*r,*newNum,*newDen;
        int pord,j,i,count,nord;
        r=(float*)malloc(sizeof(float));
        newNum=(float*)malloc(sizeof(float)*3);
        newDen=(float*)malloc(sizeof(float)*3);
        count=0;
        //m e' l'indice che mi indica da quale notch devo partire! Sempre 0. C'e' un
        //solo notch I precedenti sono a zero!
        nord=ord;
        newNum[0]=1.0;
        newNum[1]=0.0;
        newNum[2]=0.0;
        newDen[0]=1.0;
        newDen[1]=0.0;
        newDen[2]=0.0;
        //Imposto count a zero poiche' se elimino dalla convoluzione un notch, con
        //count, e questa e' la prima,
        //non considero l'ultimo notch che potrebbe essere diverso da zero
```

```
        i = 0;
        if(!(peak[pos+i].getFc()==0.0 && peak[pos+i].getG()==0.0 &&
                peak[pos+i].getFb()==0.0)){
                if(count==0){
                        //primo notch che analizzi
                        vo = pow(10,hrefl_tilde_interp(pos,i).getG()/20);
                        ho = vo - 1;
                        l = -cos(2*PI*hrefl_tilde_interp(pos,i).getFc() / sys_getsr() );
                        k= ( tan(PI*hrefl_tilde_interp(pos,i).getFb()/sys_getsr())-vo ) /
                        ( tan(PI*hrefl_tilde_interp(pos,i).getFb()/sys_getsr())+vo );

                        newNum[0]=1+(1+k)*ho/2.0;
                        newNum[1]=l*(1-k);
                        newNum[2]=-k - (1+k)*ho/2.0;
                        newDen[0]=1;
                        newDen[1]=l*(1-k);
                        newDen[2]=-k;
                        count=1;
                }
        }

        num=newNum;
        den=newDen;
        ord=nord;

        hrefl_tilde_filter(ord, num, den, ord, x, y);
}
```

A.2.2 Constanti e variabili importanti

Le costanti sono simili a quelli per *hres~* che sono quindi omesse.

```
#define PI 3.14159265358979323846
#define Nntch 1 //numero di notch per classe
#define Nang 17 //numero di intervalli da -45 a +45



// Pure Data 'class' declaration
static t_class *hrefl_tilde_class = NULL;

// struct definition for hrefl_tilde class
typedef struct _hrefl_tilde
{
        t_object x_obj;                 // Pd object struct
        t_float default_input;          // for MAINSIGNALIN
        //tipo del filtro
        t_int hrefl_type;
        //ID del soggetto
        t_int subject_id;
} t_hrefl_tilde;
```

BIBLIOGRAFIA

[1] M. Geronazzo, S.Spagnol, and F. Avanzini. Customized 3D sound for innovative interaction design. In Proc. Italian ACM SigCHI Conf. on Computer-Human Interaction (CHItaly'2011), Alghero, Sep. 2011

[2] F. Avanzini and G. De Poli. Algorithms for Sound and Music Computing. 2010. Capitolo 4
Disponibile all'indirizzo http://smc.dei.unipd.it/education.html.

[3] M. Geronazzo, S.Spagnol, and F. Avanzini. A Head-Related Transfer Function Model for Real-Time Customized 3-D Sound Rendering. In Proc. IEEE/ACM Signal-Image Technology and Internet-Based Systems (SITIS'11), Dijon, Nov. 2011

[4] J. Blauert. Spatial Hearing: Pyschophysics of Human Sound Localization. MIT
Press, 2nd edition, 1996.

[5] R. Aiello and A. Sloboda. Musical Perceptions.Oxford University Press 1994.