UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

TESI DI LAUREA TRIENNALE IN INGEGNERIA
DELL'INFORMAZIONE

MODELLO STRUTTURALE DI TESTA PER IL RENDERING AUDIO 3D NEL CAMPO VICINO

Relatore: Federico Avanzini

Correlatore: Michele Geronazzo

Laureando: *Lorenzo Visentin*

ANNO ACCADEMICO 2011/2012

Alla mia famiglia.

The LaTeX template for this thesis is available at
www.dei.unipd.it/~avanzini

Indice

Sommario

La disponibilità di dispositivi hardware sempre più potenti ha reso possibile di recente la fruizione su larga scala di applicazioni che si occupano della riproduzione di audio tridimensionale. È su questo scenario che il lavoro svolto si pone come un tentativo di integrare modelli matematici già esistenti, facendo uso di strumenti basilari quali le tecniche binaurali e le HRTF (Head Related Transfer Function). In particolare è stato realizzato un filtro a basso costo computazionale che tiene conto del contributo che la testa ha sulla trasformazione delle onde sonore per sorgenti poste a distanza ravvicinata, ovvero nel near field. In tale situazione, infatti, il fronte di un'onda acustica non può più essere approssimato con un piano, rendendo necessaria l'introduzione di un modello più raffinato e completo. Lo studio che è stato effettuato rappresenta un primo passo in questa direzione ed apre la strada ad ulteriori perfezionamenti e alla ricerca di possibili applicazioni.

Prefazione

Per comprendere appieno la nozione di *audio 3D* è necessario tornare indietro nel tempo di oltre un secolo. Erano infatti i primi anni del Novecento quando John Strutt, meglio noto come Lord Rayleigh, aveva tra i primi cominciato a studiare in modo sistematico la percezione del suono nello spazio. Nel libro *On our perception of sound direction* il fisico aveva infatti esposto quella che viene ancora oggi considerata la teoria fondamentale sulla localizzazione del suono: la cosiddetta *Duplex Theory*.

Negli anni successivi sono proseguiti, conseguendo numerosi successi, gli esperimenti e le ricerche in questo ambito, fino a giungere ai giorni nostri, allorché la disponibilità di strumenti hardware e software sempre più potenti e raffinati ha aperto nuove strade da esplorare e nuove sfide da intraprendere.

Su questo scenario si pone il lavoro svolto durante la realizzazione di questa tesi. In particolare si è fatto uso delle *tecniche binaurali* per l'ascolto del suono in cuffia; si è cercato cioè di simulare un segnale audio che arriva al timpano dell'orecchio in condizioni naturali di ascolto.

Prima di tutto è però indispensabile citare un fatto di notevole importanza: le parti anatomiche dell'uomo comportano la distorsione delle onde sonore da cui vengono investite a causa dei fenomeni di riflessione e diffrazione. Le *Head Related Transfer Function*, abbreviate con l'acronimo *HRTF*, sono le funzioni di trasferimento che racchiudono in sé l'informazione riguardo questi comportamenti fisici.

Un concetto chiave, che costituisce l'argomento principale di questa tesi, è quello di *near field*; queste parole vengono utilizzate quando si vuol far riferimento al campo acustico in prossimità dell'ascoltatore. Quando la sorgente è posta a distanza ravvicinata, infatti, il fronte di un'onda sonora non può più essere approssimato con un piano, rendendo necessaria l'introduzione di modelli matematici più raffinati e completi.

Nel capitolo 1 verrà descritto come un'onda si propaga, viene trasformata in seguito all'interazione con altri oggetti e viene percepita dal nostro sistema uditivo. La trattazione teorica che sta alla base del lavoro svolto, che è stata sviluppata da alcuni componenti del Sound and Music Computing Group, verrà illustrata con dovizia di particolari nel capitolo 2. Il capitolo 3 sarà invece dedicato a Pure Data, il software che è stato utilizzato durante i test per verificare mediante ascolto diretto la corrispondenza alle aspettative del suono ottenuto in seguito all'elaborazione. Nel capitolo 4 verrà analizzata la realizzazione in linguaggio C++ dei filtri e degli algoritmi trattati nei capitoli precedenti.

Il capitolo 5 conclude la tesi presentando le considerazioni finali e delineando i possibili perfezionamenti futuri di quanto acquisito e sviluppato finora.

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Il suono nello spazio

In questo capitolo verranno esposti il meccanismo attraverso il quale l'apparato uditivo dell'uomo percepisce un suono come localizzato in una ben determinata posizione nello spazio e qualche cenno sulla sintesi e sull'utilizzo delle HRTF (Head Related Transfer Function)[1]. In particolare nel paragrafo 1.1 si vedranno i concetti di ITD e ILD, nonché il modo con cui viene modellata la testa. Nel paragrafo 1.2 si parlerà con maggior dettaglio delle HRTF. Nell'1.3 sarà la volta degli algoritmi per il rendering 3D. Il paragrafo 1.4 chiude il capitolo introduttivo anticipando quali degli strumenti qui presentati saranno utilizzati in seguito.

1.1 Percezione del suono nello spazio: ITD e ILD

É noto dalla letteratura che le parti anatomiche dell'uomo interferiscono con le onde sonore emesse da una sorgente acustica principalmente a causa dei fenomeni di riflessione e diffrazione. In questo primo paragrafo si concentrerà in particolare l'attenzione sul contributo dato dalla testa, ma si farà un cenno anche su quelli dovuti all'orecchio esterno e a busto e spalle.

La particolare forma dell'orecchio esterno comporta l'insorgere di fenomeni di interferenza costruttiva e distruttiva che provocano l'amplificazione del suono associato a certe frequenze e l'attenuazione del suono relativo ad altre; inoltre questi comportamenti dipendono dalla direzione da cui provengono le onde acustiche. Il busto e le spalle invece contribuiscono introducendo nuove riflessioni ed oscurando parzialmente la testa dalle onde sonore provenienti dal basso. Spostando verticalmente dall'alto verso il basso un'ipotetica sorgente, si può constatare come al di sotto di una certa altezza le riflessioni scompaiano e cominci a manifestarsi l'effetto di mascheramento da parte del busto. Il contributo di busto e spalle non è rilevante quanto quello introdotto dall'orecchio esterno; tuttavia non va trascurato perché diventa significativo alle basse frequenze. Poiché gli effetti acustici dell'orecchio esterno si fanno sentire invece soprattutto alle alte frequenze, i due contributi si possono considerare in prima approssimazione complementari tra loro.

[1]Le informazioni riportate in questo capitolo sono per gran parte tratte da [1].

La conformazione della testa è tale da causare l'insorgere di due fenomeni:

- Un'onda acustica può giungere ad un orecchio in un istante leggermente diverso rispetto a quando arriva all'altro, a causa della finita velocità del suono e della distanza non nulla che separa le due orecchie. Usando una terminologia più precisa, si può dire che la testa introduce un *Interaural Time Difference (ITD)*.
- L'intensità del suono percepita da un orecchio può essere diversa da quella percepita dall'altro, dato che la testa agisce come un ostacolo che oscura parte del suono. In questo caso si parla di *Interaural Level Difference (ILD)*.

Per ragioni di semplicità nelle trattazioni e negli sviluppi teorici spesso si introduce un modello semplificato di testa sferica e la sorgente sonora viene considerata ad una distanza infinita, in modo tale che le onde sonore che giungono alla testa siano approssimabili con onde piane. In tal caso una formula valida per il calcolo dell'ITD potrebbe essere la seguente:

$$ITD \sim \frac{a}{c}(\theta + sin\theta), \tag{1.1}$$

dove a è il raggio della testa e θ l'angolo compreso tra la semiretta che parte dal centro della testa e prosegue frontalmente, e la semiretta che parte dal centro della testa e prosegue nella direzione della sorgente.

Mentre è ragionevole supporre l'ITD indipendente dalla frequenza, lo stesso non vale per l'ILD in quanto le dimensioni della testa non sono tali da attenuare sensibilmente un suono avente una frequenza bassa e quindi lunghezza d'onda molto maggiore rispetto al raggio della testa.

Il contributo della testa nella manipolazione delle onde sonore dovuto alla diffrazione può essere approssimato e descritto analiticamente dalla seguente risposta in frequenza:

$$H_{sphere}(\rho, \mu, \theta_{inc}) = -\frac{\rho}{\mu} e^{-i\mu\rho} \sum_{m=0}^{+\infty} (2m+1) P_m(cos\theta_{inc}) \frac{h_m(\mu\rho)}{h'_m(\mu)}, \tag{1.2}$$

dove ρ, definito come r/a è la distanza normalizzata, con $r > a$ distanza dal centro della sfera, $\mu = \omega a/c$ è la frequenza normalizzata, θ_{inc} è l'angolo compreso tra la semiretta che parte dal centro della testa e prosegue frontalmente, e la semiretta che parte dal centro della testa e passa per il punto d'osservazione posto sulla superficie della testa, e P_m ed h_m sono rispettivamente il polinomio di Legendre di ordine m e la funzione sferica di Hankel.

Nei paragrafi seguenti si vedrà una versione ulteriormente approssimata di questa risposta in frequenza.

1.2 Sistemi di coordinate e HRTF

Per rappresentare un punto che giace su una testa è possibile usare le coordinate sferiche, nelle quali l'angolo di azimuth e l'elevazione sono indicati rispettivamente dagli angoli θ e ϕ. In particolare è possibile esprimere le coordinate sferiche in due modi distinti:

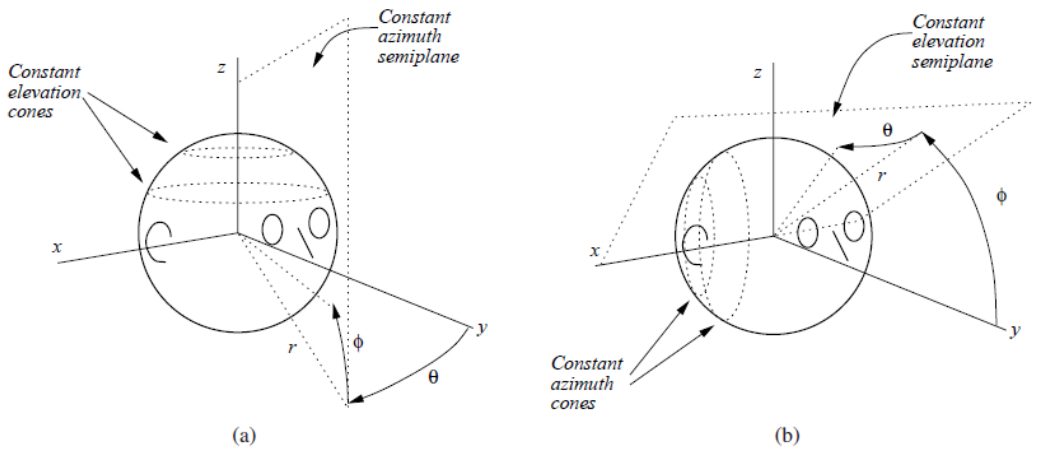

Figura 1.1: Sistemi di coordinate sferiche: (a) verticali polari, (b) interaurali polari.

- *Sistema di coordinate verticali polari:* in questo caso l'azimuth è rappresentato dall'angolo tra il piano yz ed il piano verticale contenente la sorgente, mentre l'elevazione è l'angolo a partire dal piano xy.
- *Sistema di coordinate interaurali polari:* qui l'elevazione è misurata come l'angolo tra il piano xy ed il piano contenente la sorgente e l'asse x, mentre l'azimuth è l'angolo dal piano yz.

Da notare che in quest'ultimo caso i punti aventi stesso azimuth, che vanno a formare dei coni, condividono gli stessi valori di ITD e ILD, essendo la testa sferica. Tali sistemi di coordinate sono ben illustrati in Figura 1.1.

Una volta fatta questa premessa, occorre introdurre ulteriori definizioni [2]. Una misura molto comune per descrivere un segnale è il suo livello *root mean square (RMS)*, definito semplicemente come la radice quadrata della potenza. Nel caso di segnali audio il valore istantaneo che viene assunto rappresenta una *pressione acustica*; siccome le onde acustiche viaggiano nello spazio, si può supporre che la potenza RMS sia distribuita su tutta la superficie del fronte d'onda, cosicché si possa esprimere la forza dell'onda come potenza per unità di area, nota come *intensità*. Il cosiddetto *sound pressure level (SPL)* non corrisponde esattamente all'intensità, tuttavia la sua definizione ne fa uso, così come riportato in seguito:

$$SPL = 10 \log_{10}(I/I_0) \quad (dB), \tag{1.3}$$

dove I ed I_0 sono rispettivamente l'intensità RMS del segnale ed una intensità di riferimento pari alla più piccola intensità sonora udibile dall'orecchio umano; si noti inoltre che tale misura è espressa in decibel.

Gli effetti descritti nel paragrafo precedente vengono codificati per mezzo di risposte impulsive tra la sorgente ed il timpano dell'orecchio, dette *Head Related Impulse Response (HRIR)*; le relative trasformate di Laplace vengono dette *Head Related Transfer Function (HRTF)*. Si può definire l'HRTF ad un orecchio come il rapporto, dipendente dalla frequenza, tra il *sound pressure level (SPL)* $\Phi^{(l),(r)}(\theta, \phi, \omega)$ in corrispondenza del timpano e l'*SPL* nel campo libero (*free-field*) al centro della testa $\Phi_f(\omega)$, come se l'ascoltatore non ci fosse:

$$H^{(l),(r)}(\theta, \phi, \omega) = \frac{\Phi^{(l),(r)}(\theta, \phi, \omega)}{\Phi_f(\omega)}. \tag{1.4}$$

Quando la sorgente sonora è posta ad una distanza tendente all'infinito (nella pratica si può supporre una distanza superiore ad un metro) si dice che questa si trova nel *far field*. In tal caso, il fronte delle onde acustiche può essere approssimato con un piano. Al contrario, quando si trova a meno di un metro si parla di *near field*. Siccome i suoni che percepiamo nella vita di tutti i giorni sono perlopiù provenienti da distanze elevate, è legittimo introdurre delle approssimazioni; quando però tale condizione non si verifica è opportuno utilizzare un modello più accurato. Questo è stato proprio l'obiettivo della tesi: individuare e realizzare una funzione di trasferimento semplificata che considerasse il contributo della testa nel near field, ma di questo si parlerà in seguito.

1.3 Localizzazione della sorgente e algoritmi per il rendering 3D

Come riportato nella *Duplex Theory* di Lord Rayleigh, l'ITD e l'ILD rappresentano i parametri chiave per la percezione dell'angolo azimutale di provenienza del suono. Tuttavia l'informazione fornita da queste due quantità può essere ambigua: è il caso in cui la sorgente sia posta ad un angolo di incidenza θ o $-\theta$ rispetto ad un orecchio, nell'ipotesi di testa sferica ed orecchie poste a $+90°$ e $-90°$ rispetto alla direzione frontale. In questa situazione infatti ITD e ILD assumono gli stessi valori. Proprio per questo motivo, frequente è il fenomeno della *confusione front-back*, secondo cui non si riesce a distinguere un suono proveniente dal retro o di fronte alla testa dell'ascoltatore.

Con il termine *lateralizzazione* si intende un particolare caso di localizzazione, in cui la sorgente viene percepita come se fosse presente all'interno della testa, sull'asse interaurale di congiunzione delle due orecchie; è quanto viene solitamente denominato come *inside-the-head localization*. Variando i valori di ITD e ILD si ha l'impressione che il suono si sposti lungo quest'asse immaginario.

Una sfida ancora aperta è individuare con precisione quali indizi aggiuntivi consentano l'*esternalizzazione* del suono nel caso di ascolto tramite cuffie. Tuttavia è noto che un grande contributo è dato dal riverbero.

Consentire una corretta percezione della distanza è forse l'operazione più difficile. In assenza di altre informazioni, l'*intensità* è l'indizio più utile; in condizioni anecoiche questa segue una legge di tipo proporzionale all'inverso del quadrato della distanza. Se invece è presente il riverbero, la sonorità (*loudness*), ovvero l'intensità percepita, non

varia molto al variare della distanza; la variazione del rapporto tra energia riflessa e diretta (*rapporto R/D*) sembra invece essere più significativa.

Altro fatto importante è la percezione della distanza nel caso di sorgente posta nel *near field*; in tal caso infatti le onde sonore che raggiungono l'ascoltatore sono curve e non più approssimabili con piani. Da un punto di vista qualitativo questo fenomeno corrisponde all'aumento della profondità del suono alle basse frequenze al progressivo avvicinarsi della sorgente all'ascoltatore.

Da menzionare infine gli indizi di tipo dinamico: il movimento della testa nel tempo ha di fatto un ruolo fondamentale per la localizzazione dei suoni.

Esistono varie tecniche per la manipolazione e la gestione delle HRTF; in questa tesi verranno considerati il *rendering basato su HRTF* e i *modelli strutturali*.

Nel primo caso ci si serve di HRTF ricavate sperimentalmente. Tali funzioni di trasferimento vengono ottenute mediante alcuni test per mezzo di misurazioni di risposte ad opportuni segnali utilizzando una coppia di microfoni posti all'interno delle orecchie di un ascoltatore; quest'ultimo può essere una persona reale oppure un manichino, ovvero un corpo artificiale che sintetizza le caratteristiche medie di un più vasto insieme di persone. Lo svantaggio principale di questa metodologia è dovuto al fatto che le risposte misurate variano considerevolmente da persona a persona, perciò è preferibile utilizzare un diverso set di HRTF per ciascun individuo. Dopo aver rimosso informazioni ridondanti in un set di risposte è necessario passare alla sintesi dell'HRTF. A tale scopo sono disponibili due tecniche, di cui si farà solo un cenno. Nei *modelli polo-zero* il problema si riduce all'individuazione di un filtro opportuno; lo svantaggio principale è costituito dalla complessità dei coefficienti ricavati. L'altra tecnica è rappresentata dalle *Espansioni in serie*, che hanno l'obiettivo di descrivere le HRTF come somma pesata di funzioni più semplici; una procedura statistica spesso utilizzata quando si fa uso di questa tecnica è *principal component analysis (PCA)*, la quale pone l'attenzione sulla riduzione della dimensione dell'insieme di dati pur mantenendone gli aspetti rilevanti. Anche nel caso delle espansioni in serie tuttavia resta il problema della complessità tale da renderne difficile l'impiego in un ambiente real-time.

L'idea che invece sta alla base dei *modelli strutturali* è quella di scomporre il corpo umano nelle sue componenti più rilevanti (testa, busto, orecchio esterno), le quali introducono contributi di natura diversa; per ognuna di tali parti viene poi costruita una funzione di trasferimento appropriata. Utilizzando HRTF parametrizzate da grandezze antropometriche, per esempio come il raggio della testa, la forma dell'orecchio esterno, ecc., è poi possibile adattare queste funzioni al singolo individuo per cui sono destinate. Impiegando un modello strutturale è anche possibile aggiungere il contributo della stanza nella quale si trova l'ascoltatore. Un esempio di modello strutturale che tiene conto di questi effetti è riportato in Figura 1.2.

Per quanto riguarda il contributo della testa, al fine di semplificare la suddetta risposta in frequenza H_{sphere}, è stata introdotta la seguente approssimazione del primo ordine, dove

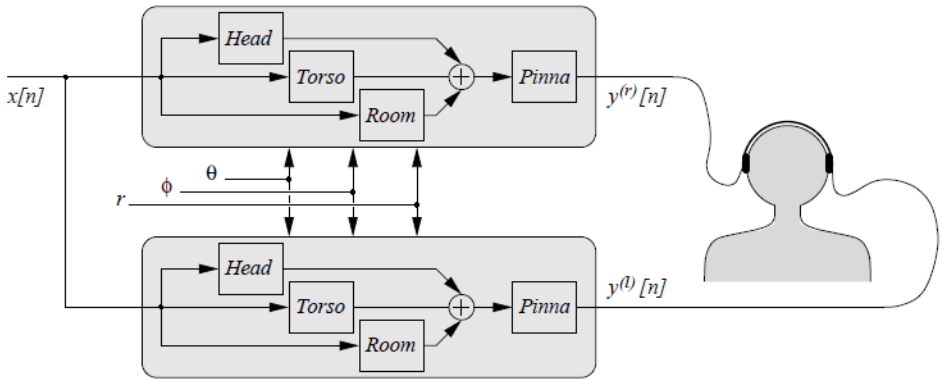

Figura 1.2: Esempio di schema a blocchi di modello strutturale.

si è considerata una sorgente infinitamente distante [7]:

$$\tilde{H}_{sphere}(\omega, \theta_{inc}) = \frac{1 + j\frac{\alpha(\theta_{inc})\omega}{2\omega_0}}{1 + j\frac{\omega}{2\omega_0}}, \quad 0 \leq \alpha(\theta_{inc}) \leq 2, \tag{1.5}$$

dove $\omega_0 = c/a$ e α è un parametro che regola la posizione dello zero e fa sì che per $\theta_{inc} = 0$ la funzione $\tilde{H}_{sphere}$ presenti modulo massimo. Solitamente α viene scelto come:

$$\alpha(\theta_{inc}) = \left(1 + \frac{\alpha_{min}}{2}\right) + \left(1 - \frac{\alpha_{min}}{2}\right) \cos\left(\frac{\theta_{inc}}{\theta_{min}} 180\right), \tag{1.6}$$

dove α_{min} e θ_{min} sono parametri euristici di controllo che talvolta assumono rispettivamente i valori 0.1 e 150°, o simili, e tutti gli angoli sono espressi in gradi. In Figura 1.3 è riportato il confronto tra il modulo di H_{sphere} e della sua approssimazione.

1.4 Strumenti utilizzati

In quest'ultimo paragrafo sarà spiegato brevemente quali degli strumenti descritti in questo capitolo verranno utilizzati. Alla luce degli aspetti positivi propri dei modelli strutturali (semplificazione dei calcoli, suddivisione del problema in sottoproblemi ciascuno legato ad una sola parte anatomica) è chiaro come nel seguito verrà adottata questa metodologia. Poiché l'intero lavoro è stato concepito in funzione della trattazione del near field, nella parte restante dell'opera non si farà riferimento a busto e orecchio esterno, ma solo alla modellazione della testa. A tale scopo si farà uso anche della risposta in frequenza semplificata riportata in formula 1.5, che tiene conto del far field.

Per quanto riguarda il sistema di riferimento impiegato, ai fini di quanto svolto l'uso del primo sistema o dell'altro è del tutto indifferente, perché nel seguito si farà riferimento solo al piano orizzontale e non si terrà conto dell'elevazione.

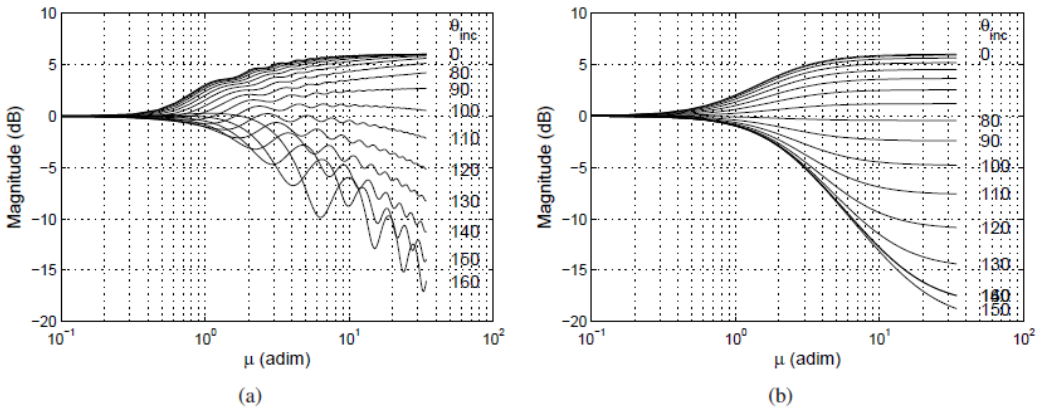

Figura 1.3: Modello per la testa: (a) H_{sphere} per $\rho \to \infty$, (b) risposta approssimata con $\alpha_{min} = 0.1$ e $\theta_{min} = 170°$.

Infine bisogna fare qualche considerazione sul sistema d'ascolto che è stato scelto, ossia le cuffie. L'ascolto in cuffia presenta alcuni svantaggi, che vengono qui elencati:

- la risposta in frequenza non è costante per tutto il range di frequenze udibili dall'uomo;
- l'uso prolungato può risultare scomodo e fastidioso.

Tuttavia le cuffie presentano due grossi vantaggi:

- eliminano il riverbero presente nello spazio circostante della stanza in cui è situato l'ascoltatore;
- permettono di convogliare alle orecchie due segnali distinti, senza la necessità di introdurre tecniche di cancellazione indispensabili invece per l'ascolto tramite altoparlanti.

Ciò premesso, si può passare alla sezione successiva, che si occupa della trattazione teorica di quanto emerso dalle indagini sul *near field*.

Capitolo 2

Modello di STF nel campo vicino

Come si è già avuto modo di accennare, la ricerca effettuata dai componenti del Sound and Music Computing Group ha avuto come obiettivo lo studio della propagazione e distorsione del suono con l'avvicinamento della sorgente sonora all'ascoltatore, in modo da riprodurne nel modo più fedele possibile gli effetti[1]. L'intento è stato quello di progettare un filtro che fosse in grado di assolvere a questa funzione garantendo una complessità tale da renderne possibile l'applicazione in ambienti real-time senza richiedere l'impiego di grandi risorse.

Questo capitolo si articola nel seguente modo: nel paragrafo 2.1 si richiameranno brevemente concetti già esposti nel capitolo precedente, in modo da presentare un inquadramento teorico specifico. Nel paragrafo 2.2 verrà descritta la procedura che ha portato alla sintesi della funzione di trasferimento appropriata; infine nel paragrafo 2.3 sarà verificata la bontà del modello individuato.

2.1 Richiami e STF

Un fatto risaputo e confermato dagli esperimenti è che, con l'avvicinarsi della sorgente all'ascoltatore, l'ITD resta pressoché invariato mentre l'ILD subisce una profonda trasformazione, specialmente a bassa frequenza, dove viene fortemente amplificato. Per questo motivo si è reso necessario introdurre una nuova funzione di trasferimento parametrizzata dalla distanza che tenesse conto di questo fenomeno. Nell'analisi che segue, come sempre, la testa sarà supposta sferica, per le semplificazioni che quest'assunzione introduce dovute alla simmetria; si parlerà perciò sempre di *Spherical transfer function (STF)* in luogo dell'acronimo più generico HRTF.

Trascurando per un attimo la variazione della distanza e supponendo che la sorgente si trovi all'infinito rispetto alla testa, è possibile impiegare la seguente STF per compendiare gli effetti di diffrazione introdotti:

$$H(\mu, \theta_{inc}) = \frac{1}{\mu^2} \sum_{m=0}^{\infty} \frac{(-i)^{m-1}(2m+1)P_m(\cos\theta_{inc})}{h'_m(\mu)}, \tag{2.1}$$

[1]Le informazioni riportate in questo capitolo sono per gran parte tratte da [3].

dove, come sempre, θ_{inc} è l'angolo d'incidenza compreso tra la semiretta che parte dal centro della testa e prosegue frontalmente, e la semiretta che parte dal centro della testa e passa per il punto d'osservazione posto sulla superficie della testa, $\mu = f\frac{2\pi a}{c}$ è la frequenza normalizzata, con a raggio della testa e c velocità del suono, e P_m ed h_m sono rispettivamente il polinomio di Legendre di ordine m e la funzione sferica di Hankel.

Reintroducendo la parametrizzazione dalla distanza, si ottiene invece la formula 1.2 ($H_{sphere}(\rho, \mu, \theta_{inc})$), in cui ρ, detta *distanza normalizzata*, è il rapporto tra distanza assoluta dal centro della testa e raggio della sfera. In Figura 2.1 vengono illustrati due esempi di moduli di funzioni di trasferimento calcolati per 19 angoli d'incidenza e per due diverse distanze normalizzate, corrispondenti a near field e far field.

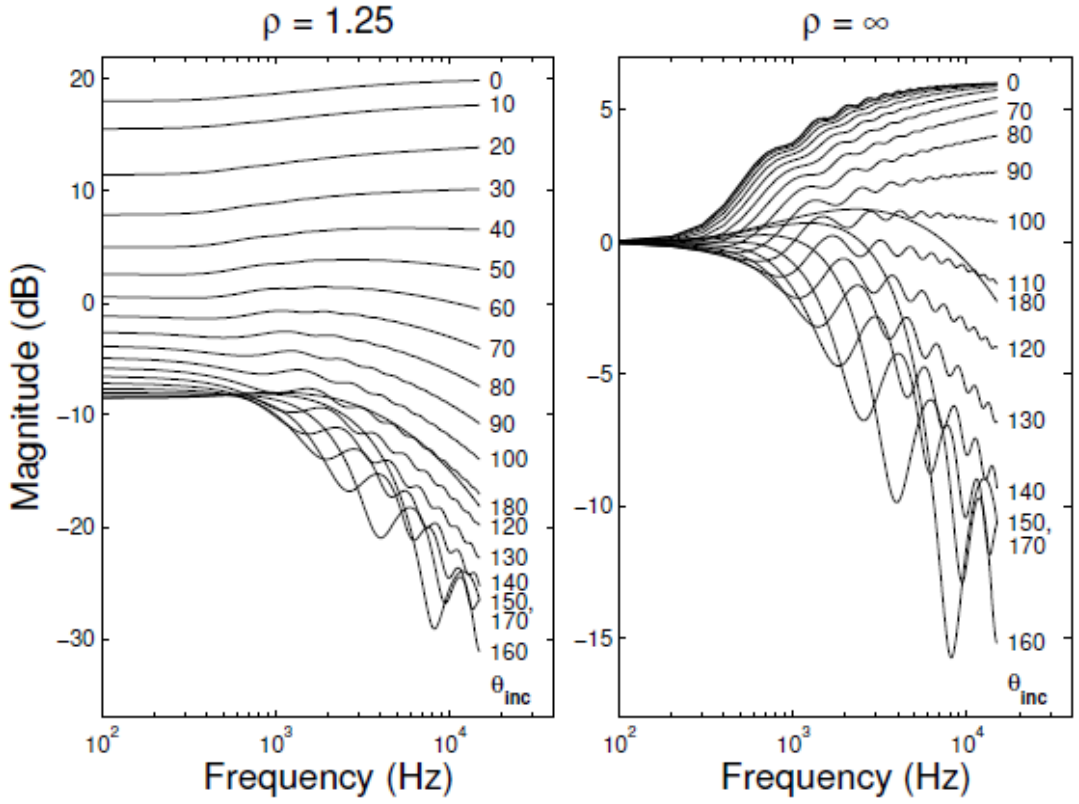

Figura 2.1: STF calcolata per $\rho = 1.25$ (*near field*) e $\rho \to \infty$ (*far field*).

In un lavoro precedente [4] gli autori hanno analizzato l'andamento delle STF utilizzando principal component analysis (PCA) ed è emerso che queste funzioni sono fortemente dipendenti dall'angolo d'incidenza, e in misura più attenuata dalla distanza; il primo obiettivo che è stato posto era perciò quello di separare l'informazione della distanza da quella restante.

2.2 Sintesi della funzione di trasferimento

Ai fini dell'analisi delle funzioni di trasferimento è stata introdotta innanzitutto una normalizzazione: la STF che tiene conto della distanza è stata divisa per quella valida nel caso di sorgente posta all'infinito:

$$H_{NF}(\rho, \mu, \theta_{inc}) = \frac{H_{sphere}(\rho, \mu, \theta_{inc})}{H_{sphere}(\infty, \mu, \theta_{inc})}. \tag{2.2}$$

In questo modo è stata ottenuta una funzione meno "frastagliata", con un andamento pressoché monotono con la frequenza, che è stata denominata *Near-Field Transfer Function (NFTF)*.

In seguito si farà sempre riferimento ad un'ipotetica testa sferica di raggio 8.75 cm; per impiegare una dimensione diversa basta scalare in modo uniforme i valori sull'asse delle frequenze.

2.2.1 Comportamento in continua

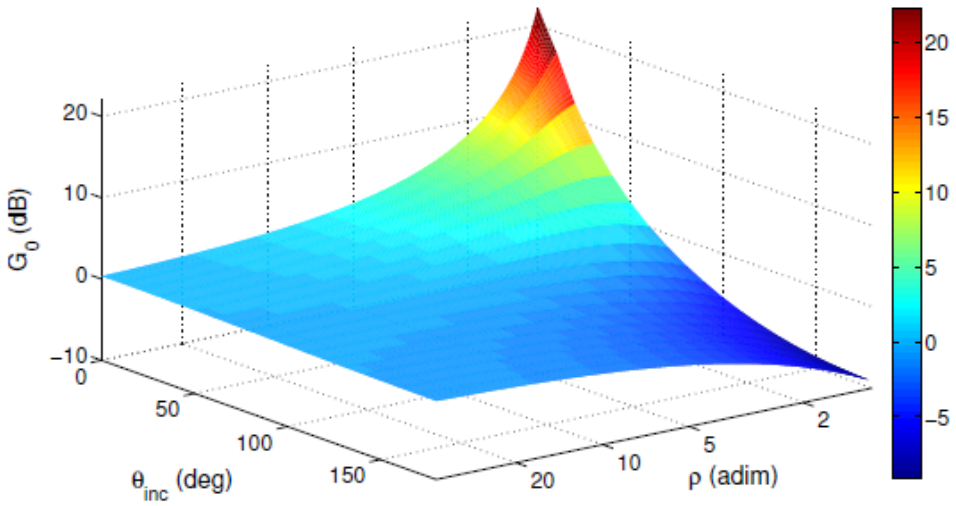

Figura 2.2: Guadagno DC della NFTF.

Le prime analisi sono state effettuate osservando come varia il guadagno DC (in continua); a tale scopo sono stati presi in considerazione 19 angoli d'incidenza compresi tra 0° e 180° ad intervalli di 10°, e 250 valori di distanza crescenti in modo esponenziale:

$$\rho = 1.15^{1+\frac{k-1}{10}}, \quad k = 1, \ldots, 250 \tag{2.3}$$

θ_{inc}	p_{11}	p_{21}	q_{11}	q_{21}	RMS [dB]
0°	12.97	-9.691	-1.136	0.219	0.0027
10°	13.19	234.2	18.48	-8.498	0.0223
20°	12.13	-11.17	-1.249	0.346	0.0055
30°	11.19	-9.035	-1.017	0.336	0.0034
40°	9.91	-7.866	-0.83	0.379	0.002
50°	8.328	-7.416	-0.666	0.421	0.0009
60°	6.493	-7.312	-0.503	0.423	0.0002
70°	4.455	-7.278	-0.321	0.382	0.0004
80°	2.274	-7.291	-0.11	0.314	0.0005
90°	0.018	-7.484	-0.13	0.24	0.0005
100°	-2.242	-8.04	0.395	0.177	0.0004
110°	-4.433	-9.231	0.699	0.132	0.0003
120°	-6.488	-11.61	1.084	0.113	0.0002
130°	-8.342	-17.38	1.757	0.142	0.0002
140°	-9.93	-48.42	4.764	0.462	0.0004
150°	-11.29	9.149	-0.644	-0.138	0.0006
160°	-12.22	1.905	0.109	-0.082	0.0003
170°	-12.81	-0.748	0.386	-0.058	0.0003
180°	-13	-1.32	0.45	-0.055	0.0002

Tabella 2.1: Coefficienti per l'equazione sul guadagno DC ed errore RMS.

e per ciascuna di queste combinazioni è stato calcolato il modulo $G_0(\theta_{inc}, \rho)$ in dB della NFTF.

Il risultato è riportato in Figura 2.2. Si nota come l'andamento è quasi sempre esponenziale, decrescente con ρ per angoli prossimi allo zero, crescente vicino ai 180°, ad eccezione per gli angoli compresi all'incirca fra 30° e 60°, dove il guadagno prima aumenta e poi diminuisce. Per questo motivo è stato scelto di approssimare il modulo con una funzione razionale del secondo ordine per ognuno dei 19 angoli d'incidenza presi in esame:

$$\tilde{G}_0(\theta_{inc}, \rho) = \frac{p_{11}(\theta_{inc})\rho + p_{21}(\theta_{inc})}{\rho^2 + q_{11}(\theta_{inc})\rho + q_{21}(\theta_{inc})}, \quad \theta_{inc} = 0, 10, \ldots, 180 \tag{2.4}$$

I coefficienti validi per ciascun angolo e gli errori RMS, sui quali si discuterà in seguito, sono riportati in Tabella 2.1. Per valutare il guadagno corrispondente agli angoli intermedi ai 19 presi in esame è stata utilizzata una semplice interpolazione lineare.

2.2.2 Comportamento al variare della frequenza

Il passo successivo è stato l'analisi per le frequenze diverse da zero, campionando l'NFTF ad intervalli di 10 Hz fino ai 15 KHz e impiegando le ben note 250 distanze. Ancora una volta è stata effettuata una normalizzazione, dividendo l'NFTF per il guadagno

DC:

$$\hat{H}_{NF} = \frac{H_{NF}(\rho, \mu, \theta_{inc})}{G_0(\theta_{inc}, \rho)}. \tag{2.5}$$

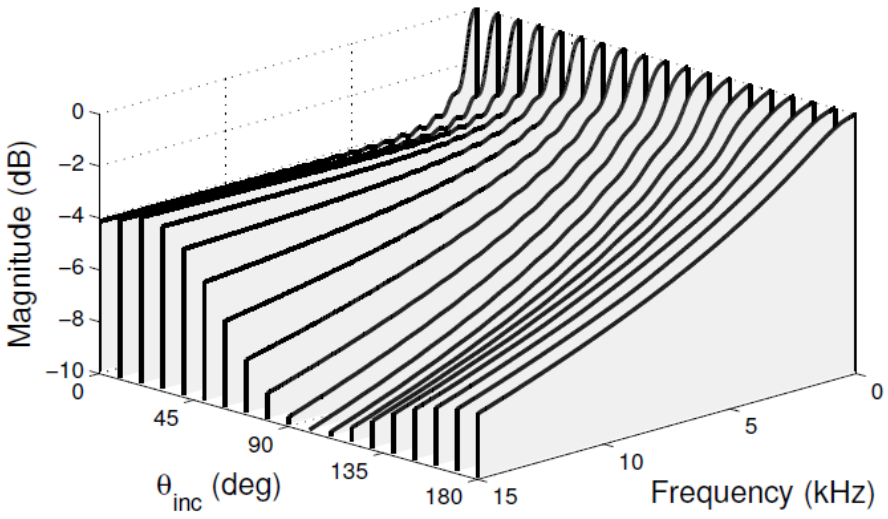

Figura 2.3: Andamento in frequenza dell'NFTF normalizzata per $\rho = 1.25$.

Il comportamento in frequenza dell'NFTF normalizzata è riportato in Figura 2.3 per un unico valore d'esempio di $\rho = 1.25$, corrispondente al near field, e per i soliti 19 angoli. Si noti come il guadagno DC rappresenti il massimo per ciascuno degli angoli e come l'andamento vari da quello di un filtro shelving per gli angoli d'incidenza ipsilaterali (prossimi a 0°) a quello di un filtro passa basso per i controlaterali (prossimi a 180°). Per simulare questo comportamento per mezzo di una funzione di trasferimento potrebbe essere una buona idea impiegare un filtro shelving per angoli piccoli, per poi passare ad un filtro passa basso a partire da una certa frequenza in poi. Tuttavia questo metodo è sconsigliato in quanto, nel passaggio per la frequenza di stacco, si sentirebbero artefatti nel suono; inoltre, un filtro passa basso introdurrebbe un'attenuazione troppo elevata per le alte frequenze.

Si è allora optato per l'utilizzo di un filtro shelving per alte frequenze del primo ordine per tutto il range di angoli d'incidenza. La legge del filtro è riportata di seguito [13]:

$$H_{sh}(z) = 1 + \frac{H_0}{2}\left(1 - \frac{z^{-1} + a_c}{1 + a_c z^{-1}}\right), \tag{2.6}$$

$$a_c = \frac{V_0 \tan\left(\pi \frac{f_c}{f_s}\right) - 1}{V_0 \tan\left(\pi \frac{f_c}{f_s}\right) + 1}, \quad V_0 = 10^{\frac{G_\infty}{20}}, \quad H_0 = V_0 - 1, \tag{2.7}$$

θ_{inc}	p_{12}	p_{22}	q_{12}	q_{22}	RMS [dB]
0°	-4.391	2.123	-0.55	-0.061	0.0007
10°	-4.314	-2.782	0.59	-0.173	0.0016
20°	-4.18	4.224	-1.006	-0.021	0.0057
30°	-4.012	3.039	-0.563	-0.316	0.0116
40°	-3.874	-0.566	0.665	-1.129	0.0199
50°	-4.099	-34.74	11.39	-8.301	0.039
60°	-3.868	3.271	-1.571	0.637	0.0151
70°	-5.021	0.023	-0.875	0.325	0.0097
80°	-6.724	-8.965	0.37	-0.083	0.0112
90°	-8.693	-58.38	5.446	-1.188	0.0179
100°	-11.17	11.47	-1.131	0.103	0.0217
110°	-12.08	8.716	-0.631	-0.12	0.0069
120°	-11.13	21.8	-2.009	0.098	0.0018
130°	-11.1	1.91	0.15	-0.401	0.0008
140°	-9.719	-0.043	0.243	-0.411	0.0014
150°	-8.417	-0.659	0.147	-0.344	0.0012
160°	-7.437	0.395	-0.178	-0.184	0.0006
170°	-6.783	2.662	-0.671	0.05	0.0006
180°	-6.584	3.387	-0.836	0.131	0.0008

Tabella 2.2: Coefficienti per l'equazione sul guadagno all'infinito ed errore RMS.

dove f_s è la frequenza di campionamento. G_∞, il guadagno in alta frequenza, è stato estratto da $\hat{H}_{NF}$ e corrisponde al guadagno in decibel dell'NFTF normalizzata a 15 KHz. f_c, la frequenza di taglio, è stata calcolata come la frequenza in corrispondenza della quale $\hat{H}_{NF}$ possiede un guadagno in dB negativo che approssima i due terzi del guadagno in alta frequenza; tale valore è stato scelto in seguito a considerazioni euristiche.

Analizzando G_∞ e f_c lungo le singole direzioni, ancora una volta si è optato per funzioni razionali del secondo ordine per approssimarne l'andamento:

$$\tilde{G}_\infty(\theta_{inc}, \rho) = \frac{p_{12}(\theta_{inc})\rho + p_{22}(\theta_{inc})}{\rho^2 + q_{12}(\theta_{inc})\rho + q_{22}(\theta_{inc})}, \tag{2.8}$$

$$\tilde{f}_c(\theta_{inc}, \rho) = \frac{p_{13}\rho^2 + p_{23}(\theta_{inc})\rho + p_{33}(\theta_{inc})}{\rho^2 + q_{13}(\theta_{inc})\rho + q_{23}(\theta_{inc})}. \tag{2.9}$$

I coefficienti per ciascun angolo e gli errori RMS sono riportati nelle Tabelle 2.2 e 2.3. Ancora una volta si è fatto uso dell'interpolazione lineare per gli angoli intermedi.

Lo schema generale impiegato per realizzare l'intera funzione di trasferimento è riportato in Figura 2.4. Si noti come il segnale in ingresso al filtro H_{sh} venga moltiplicato per il guadagno $\tilde{G}_0$ calcolato nell'equazione 2.4, producendo la funzione NFTF; moltiplicando poi quest'ultima per la funzione di trasferimento che non tiene conto della distanza, per il teorema di convoluzione si ottiene l'STF. Nel blocco denominato *Parameter extraction*

θ_{inc}	p_{13}	p_{23}	p_{33}	q_{13}	q_{23}	RMS [Hz]
0°	0.457	-0.668	0.174	-1.746	0.699	1.19
10°	0.455	0.142	-0.115	-0.01	-0.348	0.92
20°	-0.87	3404	-1699	7354	-5350	3.36
30°	0.465	-0.913	0.437	-2.181	1.188	7.01
40°	0.494	-0.669	0.658	-1.196	0.256	19.14
50°	0.549	-1.208	2.02	-1.59	0.816	30.67
60°	0.663	-1.756	6.815	-1.296	1.166	21.65
70°	0.691	4.655	0.614	-0.889	0.76	60.32
80°	3.507	55.09	589.3	29.23	59.51	29.59
90°	-27.41	10336	16818	1945	1707	36.16
100°	6.371	1.735	-9.389	-0.058	-1.118	4.54
110°	7.032	40.88	-44.09	5.635	-6.18	2.53
120°	7.092	23.86	-23.61	3.308	-3.392	2.72
130°	7.463	102.8	-92.27	13.88	-12.67	2.33
140°	7.453	-6.145	-1.809	-0.877	-0.19	2.9
150°	8.101	-18.1	10.54	-2.23	1.295	5.28
160°	8.702	-9.05	0.532	-0.96	-0.023	2.15
170°	8.925	-9.03	0.285	-0.905	-0.079	3.71
180°	9.317	-6.888	-2.082	-0.566	-0.398	3.87

Tabella 2.3: Coefficienti per l'equazione sulla frequenza di taglio ed errore RMS.

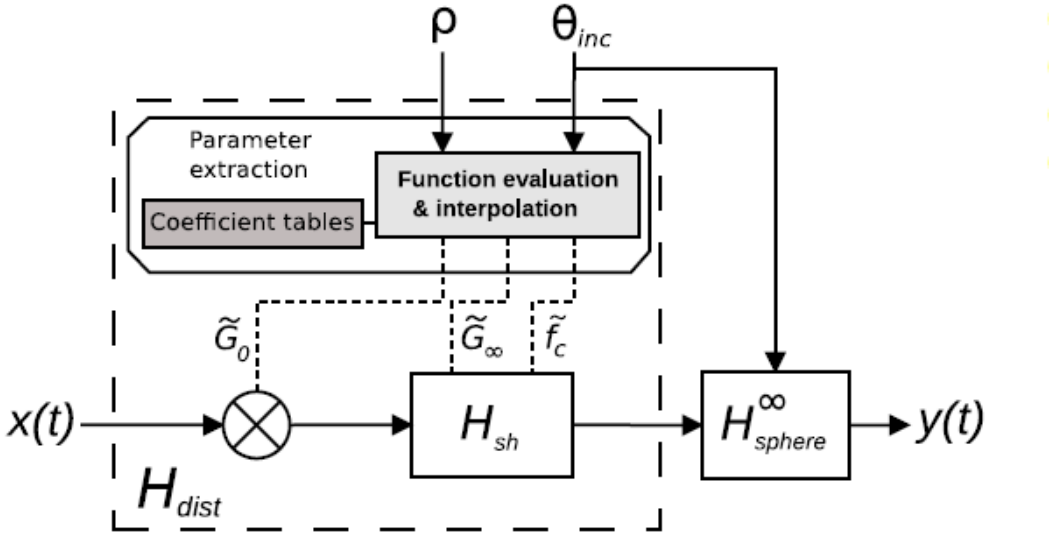

Figura 2.4: Schema a blocchi completo che rappresenta il modello che tiene conto della distanza.

vengono calcolati, previa interpolazione lineare sull'angolo, i parametri $\tilde{G}_0$, $\tilde{G}_\infty$ e $\tilde{f}_c$ che serviranno per la realizzazione dell'intero filtro.

2.3 Validità del modello e considerazioni finali

È legittimo domandarsi se le NFTF individuate racchiudano in sé le informazioni sulla distanza proprie delle HRTF; tuttavia, le registrazioni di HRTF finora sono state perlopiù effettuate nel far field, rendendo difficile trarre delle conclusioni. Ad ogni modo, l'approssimazione della testa con una sfera nel near field è stata verificata da vari studi, almeno per le basse frequenze [6].

In secondo luogo è opportuno esaminare lo scostamento tra NFTF normalizzate e relative approssimazioni mediante filtri shelving; a tale scopo sono stati confrontati i moduli di tali funzioni per tre diverse distanze. I risultati sono rappresentati in Figura 2.5.

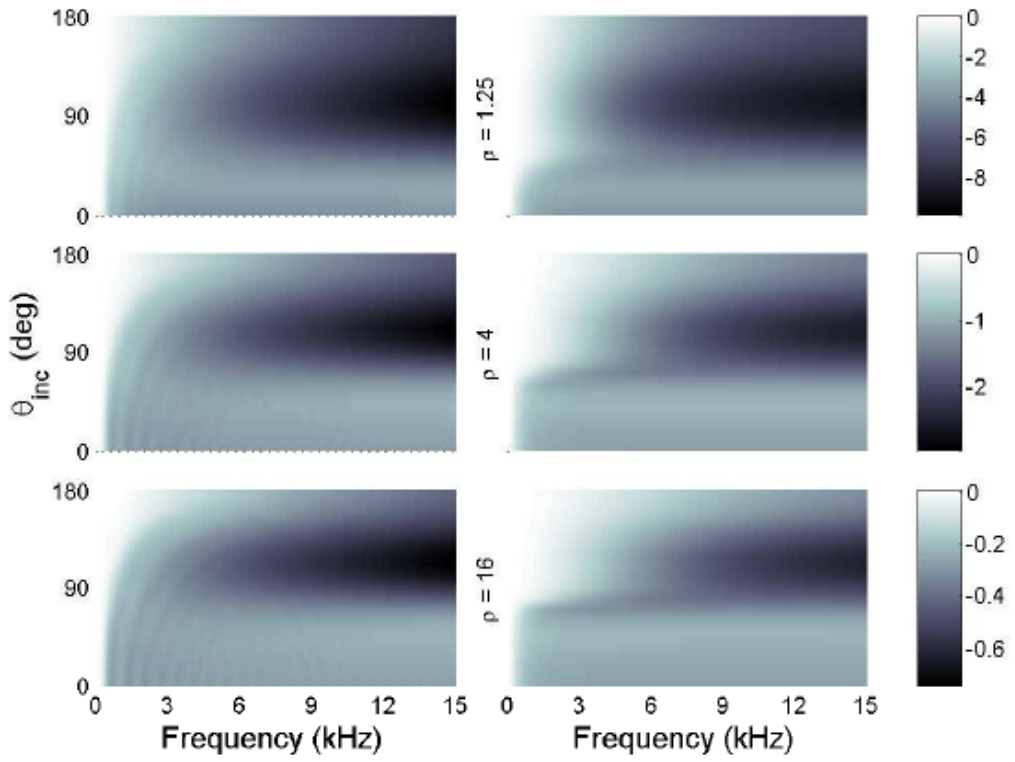

Figura 2.5: Confronto tra modulo di NFTF normalizzate (a sinistra) e di filtri shelving approssimati (a destra) per $\rho = 1.25$, $\rho = 4$ e $\rho = 16$.

Come già citato, per ciascuna delle funzioni razionali individuate per i tre parametri è stato verificato lo scostamento tra l'originale e l'approssimazione calcolando l'errore RMS (root mean square); più precisamente, tale errore è stato calcolato per ogni angolo preso in esame considerando le ben note 250 distanze. Nel caso della tabella 2.1, relativa al guadagno DC, si è riscontrata un'ottima corrispondenza dei risultati, con un errore RMS sempre minore a 0.01 dB. Lo stesso successo è stato ottenuto per la tabella 2.2, relativa al guadagno all'infinito, con un errore RMS che non supera i 0.4 dB. Infine, per la tabella 2.3, che riporta i coefficienti relativi alla funzione $\tilde{f}_c$, l'errore RMS è risultato stare al di sotto della risoluzione minima di 10 Hz con cui sono state effettuate le analisi per quasi il 70% degli angoli d'incidenza considerati.

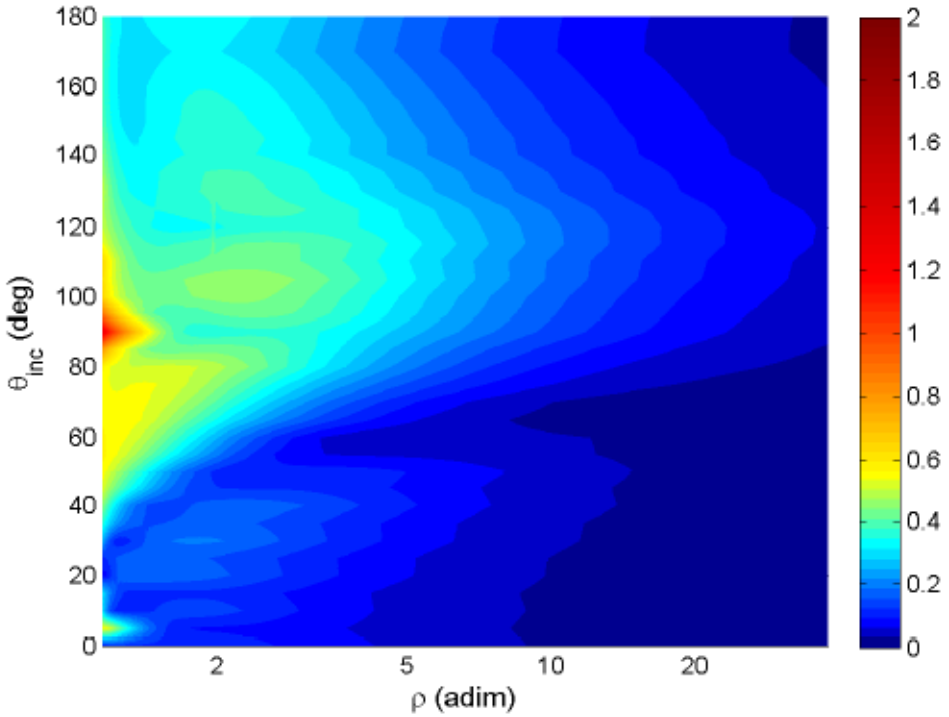

Figura 2.6: Distorsione spettrale del modello H_{dist}.

Per verificare la precisione di H_{dist} (vedere in Figura 2.4) dopo tutte le approssimazioni introdotte, è stata effettuata una misura di distorsione spettrale:

$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(20\log_{10}\frac{|H(f_i)|}{|\tilde{H}(f_i)|}\right)^2} \quad [dB], \tag{2.10}$$

dove H è la risposta originale (H_{NF} in questo caso), $\tilde{H}$ la risposta approssimata (H_{dist} in questa trattazione) e N il numero di frequenze prese in considerazione (tra 100 Hz e 15 KHz in questo lavoro). Per mezzo dell'interpolazione lineare dei parametri, l'errore è stato calcolato anche per distanze intermedie; per determinare il modulo di H_{dist} si sono impiegate le solite 250 distanze e gli angoli sono stati presi ad intervalli di 5°, sempre tra 0° e 180°.

La rappresentazione della distorsione è illustrata in Figura 2.6. Dalla figura si evince come l'approssimazione sia eccellente quasi per l'intero near field, con una distorsione sempre inferiore a 1 dB eccetto per le distanze molto piccole in corrispondenza ai 90°, com'è facile intuire considerando che in questo caso il comportamento delle NFTF normalizzate è una via di mezzo tra quello di un filtro shelving e un filtro passa basso. Si noti anche come pure per gli angoli intermedi, per i quali i parametri sono stati determinati facendo uso dell'interpolazione lineare, non vi siano scostamenti inaccettabili, tranne un piccolo salto a $\theta_{inc} = 5°$ e $\rho < 1.5$. La bontà del modello individuato impiegando l'interpolazione lineare rende ingiustificato l'uso di interpolazioni di ordine maggiore e superfluo un campionamento più denso delle funzioni.

Per concludere, è opportuno segnalare la distorsione pressoché nulla per valori di ρ elevati, che equivale a dire che H_{dist} tende a 1 per $\rho \to \infty$, confermando la validità del modello anche per il far field.

Una panoramica sui possibili sviluppi futuri e ulteriori considerazioni verranno esposti nell'ultimo capitolo, dopo aver illustrato nel dettaglio come è stato realizzato nella pratica il lavoro di ricerca descritto poc'anzi.

Capitolo 3

Pure Data

L'elaborazione del segnale audio è stata effettuata grazie al software *Pure Data*, spesso abbreviato in *Pd*, il quale ha dato la possibilità di sperimentare in prima persona la validità del modello presentato nel Capitolo 2 mediante ascolto diretto per mezzo di un paio di cuffie. Nel paragrafo 3.1 verrà dipinta una panoramica sulle finalità ed utilità di questo software; nel 3.2 si entrerà più nel dettaglio presentando una breve rassegna di *entità* ed *oggetti*. Per concludere, nel paragrafo 3.3 sarà descritto come creare nuovi oggetti con l'ausilio di *externals* scritte in linguaggio C/C++.

3.1 Introduzione a Pure Data

Come asserito nel suo sito ufficiale [9], Pure Data è un ambiente di programmazione grafico real-time per l'elaborazione di audio e video. Nato come diramazione del rinomato programma commerciale Max/MSP, Pd si è imposto come valida alternativa open source sin dagli anni '90. Sia Max che Pure Data sono stati ideati dal professore statunitense Miller Puckette, ma la forza di quest'ultimo risiede nella vasta comunità di sviluppatori sparsi in tutto il mondo che s'impegnano per ampliarne le funzionalità e metterne a frutto le potenzialità. Grazie a questo notevole contributo è stata introdotta una versione completa di librerie ed estensioni aggiuntive denominata *Pd-extended*, di cui si è fatto uso durante la realizzazione di questo lavoro. Un altro punto di forza di Pd risiede nella portabilità delle applicazioni realizzate e del programma stesso, che infatti è multipiattaforma e disponibile per sistemi operativi Windows, IRIX, GNU/Linux, BSD e MacOS X, nonché per sistemi più "esotici" ma oramai estremamente diffusi come Android e iOS per iPhone.

3.2 Patch ed entità

Una volta avviato il programma, l'ambiente che si presenta è costituito da una finestra in cui bisogna creare la propria *patch*. Una patch è un insieme di *entità* (talvolta dette *scatole*) e di collegamenti interposti tra di esse disposti in modo da ottenere la funzionalità

desiderata; per esempio, è possibile allestire un semplice lettore di file musicali, oppure generare dei suoni personalizzati introducendo effetti speciali e quant'altro. Affinché il programma restituisca in uscita del segnale audio, non si deve dimenticare di attivare il motore DSP (digital signal processor) di Pure Data spuntando la relativa casella *compute audio*. Come appena menzionato, all'interno della patch devono essere posizionate le entità, le quali possono essere di quattro tipi: oggetti, messaggi, GUI (tra cui simboli e numeri) e commenti. Gran parte delle entità è dotata di connessioni in ingresso ed uscita, denominate rispettivamente *inlets* ed *outlets*.

Gli *oggetti* sono gli elementi più importanti in una patch, in quanto permettono di manipolare dati e segnali a proprio piacimento una volta ricevuti gli opportuni ingressi. Per convenzione gli oggetti che si occupano dell'elaborazione di segnali vengono marcati nel loro nome posticipandovi il simbolo "~" (tilde), prima dell'estensione .pd; sono inoltre riconoscibili perché inlet ed outlet corrispondenti a segnali sono scuri e i cavi che ad essi si connettono sono più spessi di quelli che trasportano dati. Il primo inlet a partire da sinistra nella rappresentazione grafica dell'oggetto viene detto *caldo*, poiché alla ricezione di un messaggio, di un dato o di un segnale, produce immediatamente un'uscita; tutti gli altri inlet sono *freddi*, quindi nel momento in cui ricevono un dato, messaggio o segnale, lo inseriscono temporaneamente all'interno dell'oggetto, fino a che lo stesso non viene processato all'attivazione dell'inlet caldo [5].

I *messaggi* possono contenere stringhe di caratteri, numeri, liste o variabili e vengono attivati con un click del mouse oppure alla ricezione di un altro messaggio o di particolari azioni [5].

Un oggetto degno di essere menzionato è il cosiddetto *bang*, che consente di "innescare" un'azione; le *number box* (numeri) invece sono dei messaggi di tipo numerico.

Un'entità molto importante per l'ambiente Pd è l'oggetto *dac~*. Tale oggetto si occupa della conversione in real-time del segnale digitale in uscita, in segnale analogico riproducibile dal sistema audio. Nella rappresentazione grafica del *dac~* nell'interfaccia di Pd si nota la presenza di due ingressi per default, rispettivamente per i canali sinistro e destro. Tuttavia è possibile imporre a quest'oggetto di utilizzare un numero diverso di canali audio, a seconda della scheda audio e dell'impianto di riproduzione acustica di cui si dispone.

Per ottenere un segnale ritardato di un'opportuna quantità di tempo espressa in campioni è stato impiegato l'oggetto *delay~*.

Per finire, l'ultimo oggetto degno di nota di cui si è fatto uso è il *metro*, il quale genera una serie di bang ad intervalli di tempo prestabiliti definiti nell'argomento. Il motivo per cui è stato utilizzato quest'oggetto verrà illustrato in seguito, nel capitolo dedicato all'implementazione dell'intero lavoro.

La notevole complessità che può assumere una patch, dovuta all'impiego di una quantità considerevole di entità ed interconnessioni, può essere mitigata introducendo delle *subpatch*, ovvero delle patch all'interno della patch principale; tale processo può essere

annidato, cosicché all'interno delle subpatch possono essere create delle subsubpatch, e così via.

Per essere più precisi, esiste una distinzione tra subpatch e abstraction. Le *subpatch* appaiono come normali oggetti, al cui nome identificativo va anteposto il termine *pd*; esse possono essere utilizzate solo all'interno della patch all'interno della quale sono state create o nelle altre sue subpatch. Le *abstraction* invece, pur apparendo sempre come oggetti, vengono salvate a parte e possono essere utilizzate in qualsiasi altra patch. Affinché possa avvenire la comunicazione tra patch e sotto-patch si fa uso degli oggetti *inlet* ed *outlet*.

3.3 External in C/C++

Ora che sono state esposte le nozioni basilari per la creazione di una patch in Pd, è opportuno individuare un metodo che consenta la creazione di oggetti in modo più flessibile e articolato; per ottenere questo è sufficiente servirsi dei linguaggi di programmazione C e C++ per la scrittura dei cosiddetti *externals*. L'obiettivo di un external è quello di descrivere in modo completo e rigoroso le funzionalità che deve essere in grado di svolgere un oggetto; di tale oggetto si potrà poi fare uso come se fosse un qualsiasi altro oggetto preinstallato nella propria distribuzione di Pd.

Il software Pure Data prevede che gli external siano scritti in C; tuttavia, per venire incontro alle esigenze del programmatore e rendere meno tediosa e artificiosa la scrittura del codice, qualche anno fa è stato introdotto *Flext*, una libreria che consentiva la scrittura di codice in C++ che si integrava con Pd. Questo progetto ha reso più accessibile la scrittura di external grazie alla notevole semplificazione che introduceva. Purtroppo lo sviluppo di Flext si è interrotto parecchio tempo fa e di conseguenza la comunità di Pure Data ha ufficialmente deprecato il supporto a tale progetto.

La soluzione che è stata adottata per la realizzazione del lavoro di tesi è stata perciò quella di ricorrere ad un ambiente di programmazione C++ per mezzo del noto programma *Eclipse* [8], avendo l'accortezza di definire i nomi dei metodi con le opportune convenzioni, dettate dalla struttura nativa di Pure Data [10]. Questo ha permesso di disporre degli stessi benefici garantiti dall'uso di Flext, avendo la possibilità di introdurre nel codice i costrutti del C++, nonché di eliminare il livello aggiuntivo rappresentato da Flext.

3.3.1 Scrittura di External

Per scrivere un external in C/C++ occorre servirsi di un'interfaccia tra linguaggio C e Pure Data; la libreria *m_pd.h* fornisce le funzionalità richieste. Nel seguito verrà presentato brevemente quali sono i vari componenti che costituiscono una classe scritta

in C/C++[1].

Innanzitutto è necessario dichiarare la nuova classe e definirne lo spazio di dati (*dataspace*) [12]:

```
static t_class *myclass_class;

typedef struct _myclass {
    t_object x_ob;
} t_myclass;
```

La variabile `x_ob` è usata per memorizzare proprietà interne all'oggetto e non può essere omessa; *dopo* di essa è possibile aggiungere altre variabili a proprio piacimento.

Per avere un'interazione tra messaggi e oggetto si ricorre ad opportuni metodi, come il seguente, che viene richiamato ogniqualvolta l'oggetto creato riceve un bang:

```
void myclass_bang(t_myclass *x) {
    post("Hello world !!");
}
```

L'istruzione `post`, analoga alla funzione `printf` in C, stampa un messaggio su standard output.

Per generare la classe bisogna passare le informazioni relative ai metodi e allo spazio dei dati, come in quest'esempio:

```
void myclass_setup(void) {
    myclass_class = class_new(gensym("myclass"),
        (t_newmethod)myclass_new,
        0, sizeof(t_myclass),
        CLASS_DEFAULT, (t_atomtype) 0);
    class_addbang(myclass_class, myclass_bang);
}
```

Per questioni di brevità nella trattazione non verrà spiegato il significato di questi parametri; per maggiori informazioni si rimanda a [12].

Con le righe seguenti invece si definisce il costruttore, richiamato dalla patch di pd ogni volta che viene creato un oggetto della classe descritta nell'external:

```
void *myclass_new(void) {
    t_myclass *x = (t_myclass *)pd_new(myclass_class);
return (void *)x;
}
```

[1]Solitamente nel linguaggio C si fa riferimento al termine *struttura* in luogo di *classe*; bisogna tuttavia ricordare che il C non è un linguaggio di programmazione orientato agli oggetti, quindi il concetto di classe (o struttura) in C differisce da quello a cui comunemente si fa riferimento. Per questo motivo il codice risultante non è altrettanto elegante rispetto a quello che si otterrebbe usando il C++.

All'interno della funzione è indispensabile inserire le istruzioni che devono essere eseguite al momento della creazione.

Nel caso di classi di segnali deve essere inserito un nuovo metodo che si occupi dell'elaborazione dell'ingresso e che restituisca il segnale di uscita desiderato. Per questo motivo nel corpo del metodo `setup` devono essere aggiunte le seguenti righe:

```
class_addmethod(myclass_tilde_class,
    (t_method)myclass_tilde_dsp, gensym("dsp"), 0);
CLASS_MAINSIGNALIN(myclass_tilde_class, t_myclass_tilde, f);
```

dove ora la classe si chiama `myclass~` e non più `myclass`; la prima istruzione dichiara la presenza del metodo `dsp`, mentre la seconda afferma che esiste un ingresso di tipo segnale.

La dichiarazione della funzione `dsp` è la seguente:

```
void myclass_tilde_dsp(t_myclass_tilde *x, t_signal **sp) {
    dsp_add(myclass_tilde_perform, 5, x,
        sp[0]->s_vec, sp[1]->s_vec, sp[2]->s_vec, sp[0]->s_n);
}
```

Per maggiori informazioni si faccia sempre riferimento a [12].

Per finire si riporta un esempio per il metodo più importante della classe, quello che esegue il vero e proprio calcolo dell'uscita dato l'ingresso:

```
t_int *myclass_tilde_perform(t_int *w) {
    t_myclass_tilde *x = (t_myclass_tilde *)(w[1]);
    t_sample *in1 = (t_sample *)(w[2]);
    t_sample *in2 = (t_sample *)(w[3]);
    t_sample *out = (t_sample *)(w[4]);
    int n = (int)(w[5]);

    t_sample f_pan = (x->f_pan<0)?0.0:(x->f_pan>1)?1.0:x->f_pan;

    while (n--) {
        *out++ = (*in1++)*(1-f_pan)+(*in2++)*f_pan;
    }

    return (w+6);
}
```

Si noti in particolare come finora siano sempre state utilizzate funzioni aventi come nome quello della classe separato dal segno _ dalla funzione svolta dal metodo.

Premessa questa breve rassegna sulla scrittura di external in C/C++ si può passare al capitolo successivo, che si occupa finalmente della vera e propria implementazione.

Capitolo 4

Realizzazione

In questo capitolo si descriverà la parte più importante del lavoro svolto: verrà infatti spiegato come sono state messe in pratica le conclusioni emerse dallo studio teorico preliminare. Nel paragrafo 4.1 sarà spiegato con sufficiente dettaglio quali sono e in che modo sono stati realizzati gli external in C++. Nel paragrafo 4.2 verrà illustrato come sono state organizzate le patch contenenti gli oggetti creati tramite external. Per le considerazioni finali si rimanda all'ultimo capitolo.

4.1 Scrittura degli external

Per realizzare quanto desunto dalle ricerche condotte precedentemente, ben illustrate nel Capitolo 2, è stato fatto uso di tre classi scritte in C++:

- *head_farfield~.cpp:* questa classe implementa il filtro approssimato del primo ordine $\tilde{H}_{sphere}(\omega, \theta_{inc})$ della formula 1.5, riportato in [7], il quale *non* tiene conto della distanza;
- *head_nearfield~.cpp:* questa è la classe che realizza la funzione di trasferimento H_{dist} (vedere in Figura 2.4), che approssima $H_{NF}(\rho, \mu, \theta_{inc})$ di formula 2.2; ponendo in cascata questo filtro e quello esposto al punto precedente si ottiene la risposta complessiva, che simula $H_{sphere}(\rho, \mu, \theta_{inc})$ di formula 1.2;
- *itd.cpp:* in questo external è stato calcolato il numero di campioni di ritardo in modo da soddisfare gli effetti prodotti dall'ITD (vedere paragrafo 1.1).

Da non dimenticare che ognuno di questi oggetti è stato inserito due volte all'interno delle patch, in quanto ciascuno di essi dev'essere replicato sia per l'orecchio sinistro che per il destro. Nel seguito si analizzerà ciascuna classe singolarmente.

4.1.1 head_farfield~.cpp

Non appena un oggetto relativo a questo external viene inserito in una patch di Pd, come di consueto viene richiamato il metodo costruttore della classe; questo a sua volta

effettua una chiamata ad una funzione che si occupa del calcolo dei coefficienti della già nota risposta in frequenza $\tilde{H}_{sphere}(\omega, \theta_{inc})$. Tuttavia, poiché il calcolatore effettua i conti nel dominio del tempo discreto, si è reso necessario introdurre la *trasformata bilineare* dell'HRTF approssimata; com'è noto infatti, questa trasformata consente di convertire una funzione razionale nel dominio del tempo continuo in un'altra nel dominio discreto e viceversa, per mezzo di un cambio di variabile. Ripercorrendo il procedimento svolto, prima di tutto a partire dalla formula 1.5 si è passati dalla funzione espressa nella variabile $j\omega$ (più precisamente detta risposta in frequenza) alla stessa funzione nel dominio s (funzione di trasferimento, o trasferenza), ottenendo:

$$\tilde{H}_{sphere}(s, \theta_{inc}) = \frac{1 + \frac{\alpha(\theta_{inc})}{2\omega_0} s}{1 + \frac{1}{2\omega_0} s}. \tag{4.1}$$

Esplicitando poi il cambio di variabile $s = \frac{2}{T}\frac{z-1}{z+1}$, con $T = \frac{1}{f_s}$, e compiendo semplici passaggi, si è giunti all'espressione finale nel dominio z. Qui di seguito si riportano i conti svolti:

$$\begin{aligned} H(z) &= \frac{1 + \frac{\alpha}{2\omega_0}\frac{2}{T}\frac{z-1}{z+1}}{1 + \frac{1}{2\omega_0}\frac{2}{T}\frac{z-1}{z+1}} = \frac{2\omega_0 T(z+1) + 2\alpha(z-1)}{2\omega_0 T(z+1) + 2(z-1)} = \\ &= \frac{(2\omega_0 T + 2\alpha)z + (2\omega_0 T - 2\alpha)}{(2\omega_0 T + 2)z + (2\omega_0 T - 2)} = \\ &= \frac{(\omega_0 + \alpha f_s)z + (\omega_0 - \alpha f_s)}{(\omega_0 + f_s)z + (\omega_0 - f_s)} = \frac{\frac{\omega_0 + \alpha f_s}{\omega_0 + f_s} z + \frac{\omega_0 - \alpha f_s}{\omega_0 + f_s}}{z + \frac{\omega_0 - f_s}{\omega_0 + f_s}}, \end{aligned} \tag{4.2}$$

dove nell'ultimo passaggio i coefficienti della funzione razionale sono stati normalizzati in modo da ottenere a_0 (ovvero il coefficiente a denominatore relativo alla potenza di grado uno) uguale a 1.

L'equazione alle differenze generica relativa alla funzione di trasferimento appena stilata è la seguente:

$$a_0 y(n) + a_1 y(n-1) = b_0 x(n) + b_1 x(n-1), \tag{4.3}$$

da cui:

$$y(n) = -\left(\frac{\omega_0 - f_s}{\omega_0 + f_s}\right) y(n-1) + \left(\frac{\omega_0 + \alpha f_s}{\omega_0 + f_s}\right) x(n) + \left(\frac{\omega_0 - \alpha f_s}{\omega_0 + f_s}\right) x(n-1). \tag{4.4}$$

Nel metodo `perform`, in cui, come si ricorda, si effettua l'elaborazione del segnale in ingresso e viene restituita l'uscita, è stato realizzato il calcolo proprio mediante quest'ultima equazione alle differenze. Il codice, che fa uso dei puntatori, è riportato di seguito:

```
while (n--) {
        *out = (x->bCoeff[0]) * (*in);
        x->inOld[0] = *in++;
        for (t_int i = 1; i < x->ord; i++) {
            *out += x->bCoeff[i] * x->inOld[i];
            *out -= x->aCoeff[i] * x->outOld[i];
        }
        x->outOld[0] = *out++;

        for (t_int i = x->ord - 1; i >= 1; i--) {
            x->inOld[i] = x->inOld[i - 1];
            x->outOld[i] = x->outOld[i - 1];
        }
    }
```

dove `inOld` e `outOld` sono vettori composti da due celle ciascuno contenenti i campioni relativi agli istanti precedenti in ingresso e uscita, `n` è la lunghezza dei vettori di campioni `in` e `out`, e `ord` in questo caso vale 2.

Va evidenziato come alla classe sia stato passato come parametro direttamente l'angolo d'incidenza θ_{inc} e non θ (compreso tra direzione frontale e semiretta che interseca la sorgente sonora); così facendo non si è dovuto far ricorso a due classi distinte per gli orecchi sinistro e destro per il calcolo del suddetto angolo d'incidenza. Dell'angolo ricevuto in ingresso è stato poi calcolato il modulo e, qualora necessario, è stata effettuata una normalizzazione in modo da ottenere valori sempre compresi tra 0° e 180°. Il motivo per cui è stato sempre considerato il valore assoluto dell'angolo va ricercato nella simmetria propria della sfera con cui è stata approssimata la testa: con quest'assunzione, infatti, un'onda sonora proveniente da una sorgente posta a $+\theta_{inc}$ o $-\theta_{inc}$ produce gli stessi risultati.

4.1.2 head_nearfield~.cpp

Per quanto riguarda il filtro implementato nella classe head_nearfield~.cpp, in questo caso non si è reso necessario l'uso della trasformata bilineare, poiché il filtro determinato durante la fase di sintesi era già stato espresso nel dominio discreto. È bastato quindi riorganizzare la funzione $H_{sh}(z)$ riportata in formula 2.6 in modo da esplicitarne i coefficienti:

$$\begin{aligned} H_{sh}(z) &= 1 + \frac{H_0}{2}\left(1 - \frac{z^{-1} + a_c}{1 + a_c z^{-1}}\right) = 1 + \frac{H_0}{2}\left(\frac{1 + a_c z^{-1} - z^{-1} - a_c}{1 + a_c z^{-1}}\right) = \\ &= \frac{(2 + H_0 - a_c H_0)z + (2a_c + a_c H_0 - H_0)}{2z + 2a_c} = \\ &= \frac{(1 + \frac{H_0}{2} - \frac{a_c H_0}{2})z + (a_c + \frac{a_c H_0}{2} - \frac{H_0}{2})}{z + a_c}. \end{aligned} \tag{4.5}$$

Per l'equazione alle differenze invece si ottiene:

$$y(n) = -a_c y(n-1) + \left(1 + \frac{H_0}{2} - \frac{a_c H_0}{2}\right) x(n) + \left(a_c + \frac{a_c H_0}{2} - \frac{H_0}{2}\right) x(n-1). \quad (4.6)$$

Il metodo `perform` di questa classe è del tutto equivalente a quello dell'external precedente, con la sola differenza che all'inizio del ciclo `while` si moltiplica il segnale d'ingresso per $\tilde{G}_0(\theta_{inc}, \rho)$, in accordo al modello teorico (vedere a tal proposito la Figura 2.4):

```
*in *= x->G0;
```

Per potersi servire delle tre tabelle dei coefficienti da inserire nelle funzioni razionali di $\tilde{G}_0$, $\tilde{G}_\infty$ e $\tilde{f}_c$, i dati in esse contenuti sono stati riportati in un file di testo separati dal carattere `;`. Successivamente sono stati tutti acquisiti dal codice all'atto dell'inizializzazione mediante l'uso della classe `ifstream` e memorizzati in un vettore per ciascun parametro.

Poiché l'angolo d'incidenza non deve corrispondere necessariamente ad uno di quelli riportati nelle tabelle, bensì sono ammessi anche gli angoli intermedi, è stata fatta un'interpolazione lineare dei suddetti parametri ($\tilde{G}_0$, $\tilde{G}_\infty$, $\tilde{f}_c$) sfruttando la seguente legge generale [11]:

$$f(x) = \frac{x - x_b}{x_a - x_b} y_a - \frac{x - x_a}{x_a - x_b} y_b, \quad (4.7)$$

a cui vanno sostituite le opportune quantità: x, x_a e x_b sono rispettivamente l'angolo θ_{inc} e gli angoli, arrotondati alla cifra delle decine, immediatamente inferiore e superiore a θ_{inc}, mentre y_a e y_b sono i valori dei parametri calcolati per i due angoli precedenti.

Da notare che, come per la classe head_farfield~, sono sempre stati considerati gli angoli d'incidenza (θ_{inc}), devolvendone il calcolo a partire da θ alla patch di Pd; inoltre, ancora una volta, sono stati utilizzati sempre angoli compresi tra 0° e 180°, in quanto i coefficienti delle tabelle sono stati calcolati solo per questo intervallo.

4.1.3 itd.cpp

Questo codice non si occupa di elaborazione di segnali, ma restituisce semplicemente il numero di campioni di ritardo, in accordo alla seguente legge sull'ITD (qui il ritardo è espresso in secondi) [7]:

$$\Delta T(\theta_{inc}) = \begin{cases} -\frac{a}{c}\cos(\theta) & se\ 0 \le |\theta| < \frac{\pi}{2} \\ \frac{a}{c}\left(|\theta| - \frac{\pi}{2}\right) & se\ \frac{\pi}{2} \le |\theta| < \pi. \end{cases} \quad (4.8)$$

Il codice che realizza questo calcolo è riportato qui di seguito:

```
if (abs(x->thetaInc) < itd_deg2Rad(90.0)) {
    silence = (int) ((sys_getsr() * (- x->a / x->c) *
        cos(x->thetaInc)) + (sys_getsr() * x->a / x->c));
```

```
} else {
    silence = (int) ((sys_getsr() * (x->a / x->c) *
        (abs(x->thetaInc) - M_PI / 2)) + (sys_getsr() * x->a / x->c));
}
```

dove `silence` rappresenta il numero di campioni di ritardo e la funzione `sys_getsr()` restituisce la frequenza di campionamento f_s.

Da notare come ad entrambe le espressioni della formula 4.8 sia stata aggiunta la stessa quantità `(sys_getsr() * x->a / x->c)`. Siccome tale valore è stato sommato per entrambe le orecchie, potrebbe sembrare un'operazione superflua in quanto la differenza resta sempre la stessa. In realtà questo consente di avere sempre un numero di campioni di ritardo positivo, altrimenti si tratterebbe di un anticipo, difficilmente gestibile non disponendo del segnale per istanti futuri.

Come sempre, l'angolo è stato preso in modulo e adattato in modo da essere compreso tra 0° e 180°.

4.2 Creazione delle patch

Le classi riportate nel paragrafo precedente sono state impiegate all'interno di patch Pd opportunamente allestite. L'elenco completo è riportato qui di seguito:

- *bt_head_mono.pd:* questa è l'*abstraction* più interna, che fa uso di tutte e tre le classi precedentemente descritte;
- *bt_head.pd:* questa patch contiene a sua volta due patch *bt_head_mono.pd*, una per l'orecchio sinistro e una per il destro;
- *demo.pd:* in quest'ultima patch viene data la possibilità di selezionare il file audio da riprodurre, di impostare l'angolo e la distanza desiderate ed, eventualmente, di modificare alcuni parametri.

Siccome dovrebbe essere sufficiente una rapida occhiata per intuire le funzionalità svolte dalle patch, non verranno illustrate nel dettaglio ma ci si limiterà ad esporre qualche considerazione; ad ogni modo, le patch sono riportate nelle Figure 4.1, 4.2 e 4.3.

4.2.1 bt_head_mono.pd

Si noti come l'abstraction bt_head_mono.pd riceva come input un segnale musicale monofonico, l'angolo d'incidenza in gradi, la distanza normalizzata, il raggio della testa e i parametri α_{min} e θ_{min} già citati nel paragrafo 1.3. È presente inoltre un oggetto *metro*, il cui significato è stato anticipato nel paragrafo 3.2, a cui viene passato il parametro $500ms$. Questa entità è stata introdotta affinché continuassero ad essere richiamate le funzioni che si occupano del calcolo dei coefficienti e parametri dei filtri, nonché del numero di campioni di ritardo; infatti, tutte queste quantità dipendono dall'angolo d'incidenza θ_{inc},

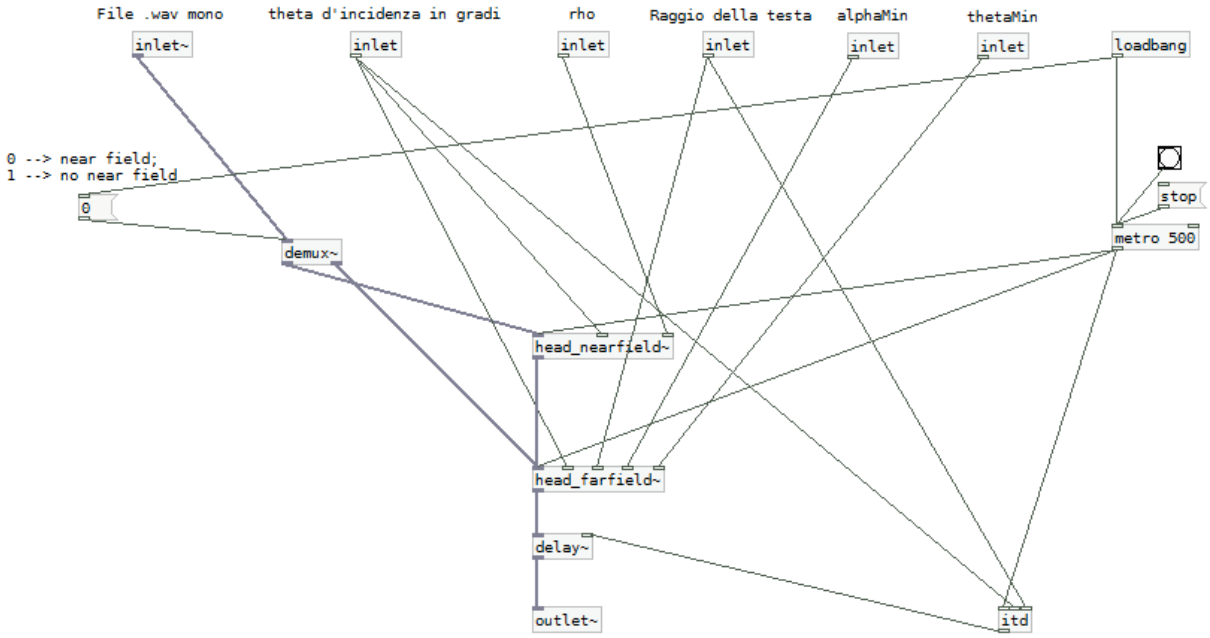

Figura 4.1: Patch *bt_head_mono.pd*.

pertanto necessitano di essere aggiornate continuamente per adeguarsi ai cambiamenti del suddetto angolo. L'oggetto *demux˜* è stato inserito per escludere, all'occorrenza, l'effetto del near field; per ottenere questo risultato quest'entità si limita a scollegare l'ingresso audio dall'oggetto head_nearfield˜ e a connetterlo direttamente a head_farfield˜, senza passare per la cascata dei due. Un altro fatto da menzionare è la presenza dell'oggetto *delay˜*, anch'esso già citato nel paragrafo 3.2, il quale lavora in coppia con il blocco itd per ritardare il segnale della quantità di campioni stabilita da quest'ultimo. Per finire, l'oggetto *loadbang* genera un bang all'apertura della patch per ognuna delle entità a cui è collegato.

4.2.2 bt_head.pd

Questa patch possiede gli stessi ingressi di quella descritta in precedenza, con la sola differenza che qui viene fornito l'angolo θ (tra $-180°$ e $180°$), non θ_{inc} (si rammenta che θ è l'angolo compreso tra la semiretta che parte dal centro della testa e prosegue frontalmente, e la semiretta che parte dal centro della testa e prosegue nella direzione della sorgente). A partire da θ vengono poi calcolati gli angoli d'incidenza per gli orecchi sinistro e destro (rispettivamente $\theta_{inc,sx} = \theta + 100$ e $\theta_{inc,dx} = \theta - 100$).

4.2.3 demo.pd

Questa è l'ultima patch che si prenderà in considerazione. Qui si effettua la scelta del file musicale e si avvia o arresta la riproduzione, facendo uso degli oggetti *readsf˜*

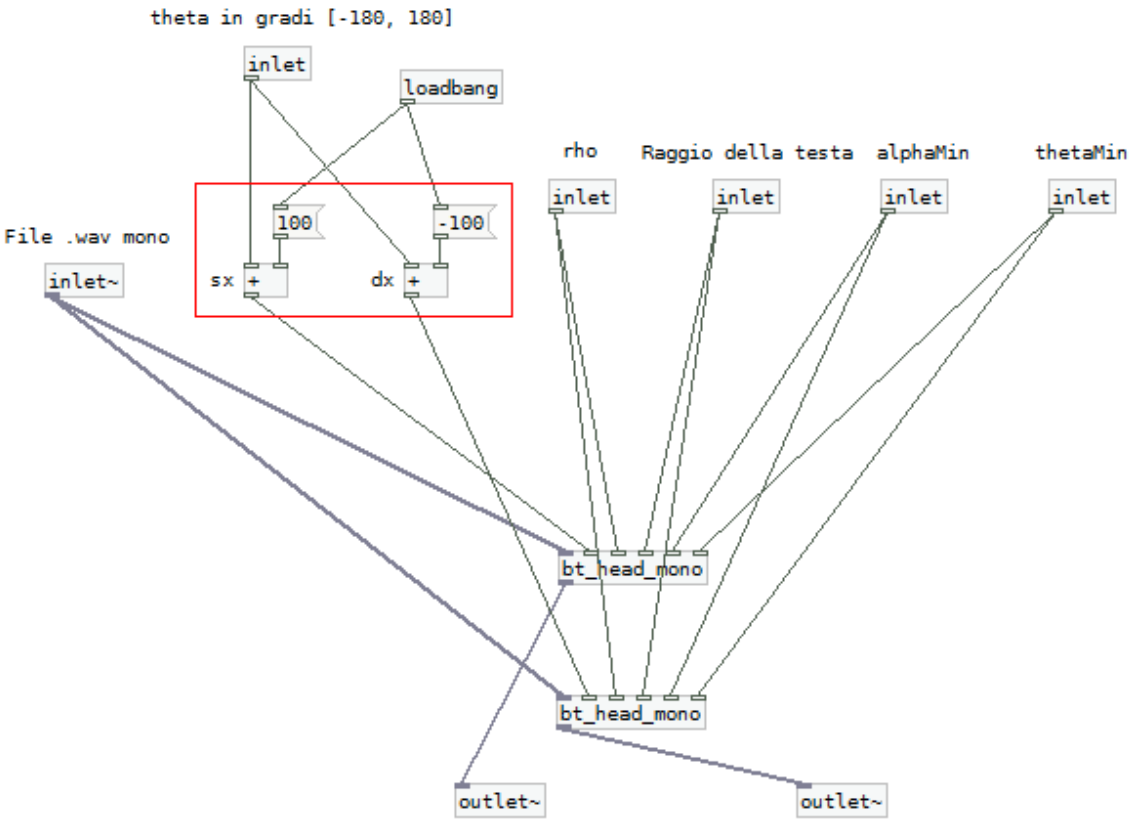

Figura 4.2: Patch *bt_head.pd*.

e *openpanel*. Sono inoltre presenti due *slider* che consentono di variare con continuità rispettivamente l'angolo θ (tra $-180°$ e $180°$) e la distanza normalizzata (tra 1, corrispondente ad una distanza di 8.75 cm per una testa di pari raggio, e 34, corrispondente a circa 3 metri); per finire, è data la possibilità di modificare i valori di raggio della testa[1] a, α_{min} e θ_{min}. L'uscita audio globale, generata dai blocchi descritti in questo capitolo connessi come illustrato nelle figure, viene infine passata all'oggetto *dac~* (vedere sezione 3.2), che la manda a sua volta alla scheda audio del computer.

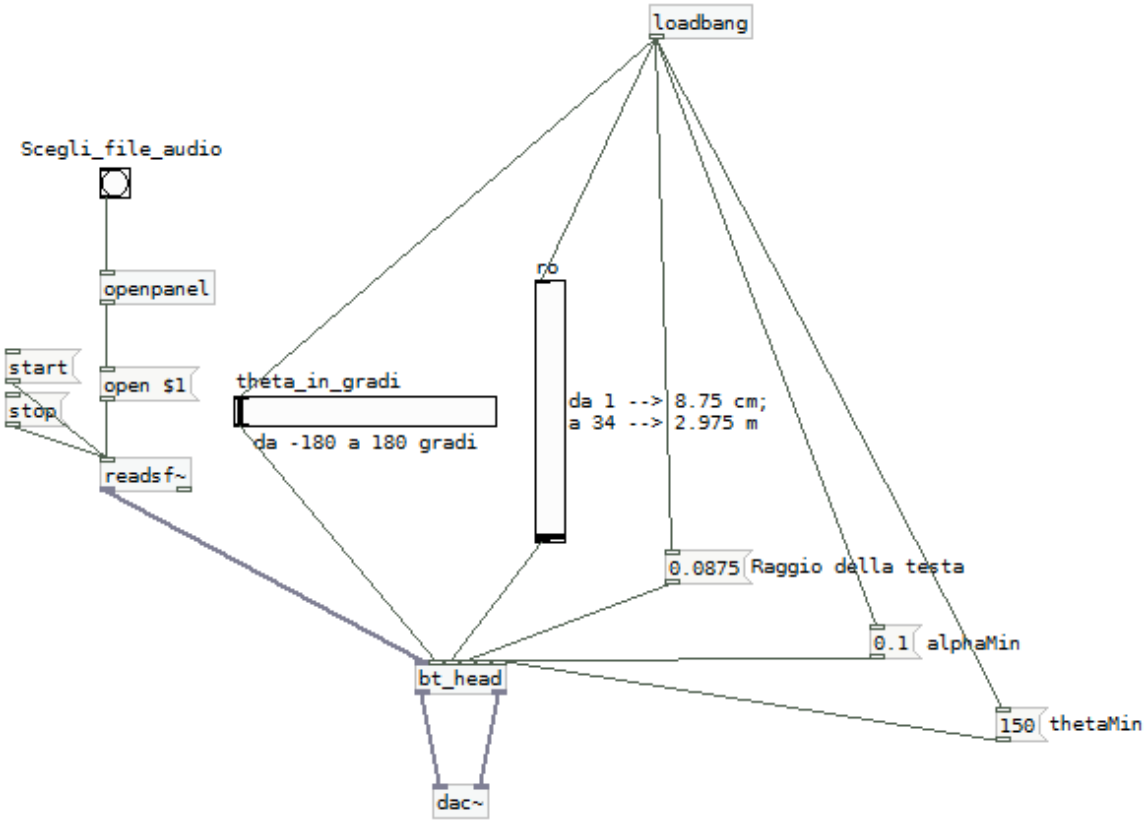

Figura 4.3: Patch *demo.pd*.

I risultati della prova d'ascolto mediante gli external e le patch sopraccitati verranno illustrati nell'ultimo capitolo.

[1] Non ancora implementato per il near field.

Capitolo 5

Considerazioni e sviluppi futuri

In questo capitolo saranno esibite alcune considerazioni e si trarranno le conclusioni sulla parte teorica trattata nel Capitolo 2 e sull'implementazione del Capitolo 4. Più precisamente, nel paragrafo 5.1 verranno presentate le conclusioni per il modello teorico, mentre nel 5.2 si parlerà dell'esperienza di ascolto con Pure Data e delle migliorie che è possibile apportare alla realizzazione.

5.1 Conclusioni sul modello teorico

Come si evince dal grafico che rappresenta la distorsione spettrale del filtro H_{dist} (vedere Figure 2.4 e 2.6), i risultati che sono stati ottenuti sono buoni per tutto il range di distanze e di angoli d'incidenza, a parte in qualche zona nel caso di sorgente estremamente vicina al soggetto (vedere il paragrafo 2.3). Tuttavia, per avere una conferma definitiva della bontà del modello è indispensabile un'esperienza di ascolto diretta; di questo fatto si parlerà più approfonditamente nel paragrafo 5.2. Tornando alla sintesi della funzione di trasferimento, si potrebbe contestare che l'impiego di sistemi LTI diversi dal filtro shelving del primo ordine che è stato considerato potrebbe portare ad approssimazioni ancora migliori. A tale scopo filtri shelving di ordine superiore o di filtri passa basso dotati di controllo dell'inclinazione potrebbero essere delle valide alternative [3]. È indispensabile poi fare delle considerazioni sull'ITD: infatti, con l'assunzione che l'ITD non cambi col variare della distanza, dovrebbe essere necessario introdurre un ulteriore filtro passa tutto (*all-pass filter)* che compensi l'effetto che la fase del filtro precedentemente impiegato ha sull'ITD. Infine, è opportuno ricordare che la bontà della realizzazione è determinata per gran parte anche dal modello scelto per il filtro che non tiene conto della distanza. Ancora una volta, infatti, l'uso di filtri di ordine superiore potrebbe mitigare ulteriormente lo scostamento dai risultati sperimentali.

5.2 Considerazioni sull'implementazione

Durante le prove d'ascolto effettuate usando Pure Data, si è potuto constatare chiaramente l'aumento dell'effetto di amplificazione impostando valori di distanza normalizzata sempre più piccoli, specialmente nello spettro delle basse frequenze, come riscontrato in precedenza dalle analisi teoriche. L'esperienza d'ascolto è stata complessivamente positiva: la combinazione tra il filtro modellato per tener conto del contributo acustico nel near field e il blocco che calcola i campioni di ritardo per l'ITD è risultata vincente. Tuttavia, il notevole guadagno complessivo determinato dall'oggetto head_nearfield~ ha consentito l'ascolto solo di file musicali registrati con un livello sonoro molto basso. L'ascolto di normali brani musicali ha invece permesso di constatare come, impostando la distanza su valori sempre più piccoli, l'audio venga riprodotto con evidenti scatti e con una sorta di fischio di sottofondo; per di più in questo caso l'ampiezza del segnale viene amplificata a tal punto da costringere ad impostare il volume in uscita del sistema ad un livello prossimo allo zero. La soluzione più plausibile potrebbe essere quella di ridurre l'amplificazione di un determinato fattore moltiplicativo; in ogni caso sarebbe opportuno verificare la corrispondenza tra i risultati sperimentali ricavati durante la fase di ricerca e quelli ottenuti con l'implementazione.

Una possibile obiezione potrebbe essere sollevata in merito all'uso dell'oggetto metro, che, come si ricorda, serviva per lanciare dei bang ad intervalli periodici, per mezzo dei quali si andavano a richiamare le funzioni per il calcolo di coefficienti, parametri e ITD. Infatti tali metodi vengono richiamati in continuazione anche se l'angolo non è stato modificato nel frattempo, comportando uno spreco di risorse che, pur non essendo molto rilevante, sarebbe sicuramente più opportuno evitare. La soluzione più ovvia consiste nell'inserimento di un'istruzione di controllo, che dovrebbe essere eseguita prima del ciclo while contenuto nel metodo perform (sia per head_farfield~ che per head_nearfield~). Tale istruzione dovrebbe confrontare l'angolo che era stato impostato precedentemente con l'angolo al momento della nuova chiamata al suddetto metodo perform; se questi dovessero essere diversi, allora sarebbe necessario richiamare le funzioni che aggiornano coefficienti, parametri, ecc. Resta tuttavia da verificare il comportamento di questa soluzione nel caso in cui l'angolo venga modificato con continuità spostando lo slider nella patch Pd; in tal caso, infatti, le funzioni di calcolo sopraccitate verrebbero richiamate un numero considerevole di volte, comportando un possibile degrado nelle prestazioni.

Altra questione è l'utilità dell'oggetto demux~, il quale era stato originariamente inserito per consentire la riduzione dell'onere computazionale nel caso di sorgente posta nel far field, poiché in tal caso sarebbe superfluo l'impiego del filtro che tiene conto del contributo nel campo vicino. Purtroppo, però, escludere l'oggetto head_nearfield~ con questa tecnica non ha portato ad alcun beneficio, dato che tutte le funzioni perform continuano ad essere richiamate a prescindere dal fatto che al relativo oggetto venga assegnato o meno un segnale in ingresso. Ad ogni modo un possibile miglioramento potrebbe essere quello di escludere automaticamente, e non a mano, il blocco head_nearfield~ non appena la distanza supera il metro e si passa quindi nella regione del far field.

Bisogna infine menzionare un problema che si è verificato durante l'ascolto in deter-

minate circostanze: disponendo e collegando le entità nelle patch in determinati modi può succedere che, impostando lo stesso angolo d'incidenza prima per un orecchio e poi per l'altro, non si ottenga lo stesso guadagno come invece ci si aspetterebbe. Si è poi riscontrato come l'uso dell'oggetto demux~ consenta di eliminare il problema. L'ipotesi più accreditata è che il motivo di questa discrepanza sia da ricercarsi nell'errata gestione dei flussi audio mono e stereo nella patch, a cui il demux~ potrebbe aver posto rimedio.

Passando invece a discutere degli sviluppi futuri di quanto assimilato e messo in pratica finora, si potrebbe osservare innanzi tutto che la trattazione teorica è stata condotta dopo aver stabilito un valore fisso per la dimensione della sfera che approssima la testa; bisognerebbe verificare teoricamente e sperimentalmente se i risultati ottenuti trovano riscontro anche per valori differenti di raggio.

Un ulteriore passo avanti potrebbe anche essere l'integrazione del progetto con un sistema di head tracking che rilevi la posizione della testa, in modo da permettere all'ascoltatore di muoversi e di sfruttare gli indizi dinamici per la localizzazione del suono (vedere il paragrafo 1.3). Infine, si potrebbe ottenere un notevole miglioramento completando lo schema che è stato realizzato con altri filtri che modellino il contributo dell'orecchio esterno, fondamentale tra l'altro per la percezione dell'elevazione, nonché del busto, i cui effetti compaiono soprattutto alle basse frequenze.

Per concludere la trattazione, va ribadito come l'esperienza d'ascolto finale abbia presentato risultati incoraggianti, lasciando intravedere buone prospettive per ulteriori sviluppi e nuove applicazioni. L'entusiasmo per tali conquiste rappresenterà forse in futuro la motivazione primaria che consentirà di apportare ulteriori progressi, aprendo la strada a nuovi modi di concepire il suono e di farne uso quotidianamente.

Bibliografia

[1] F. Avanzini, *Algorithms for Sound and Music Computing, Capitolo 4*, 2009, reperibile all'indirizzo `http://smc.dei.unipd.it/education.html`.

[2] F. Avanzini e G. De Poli, *Algorithms for Sound and Music Computing, Capitolo 1*, 2009, reperibile all'indirizzo `http://smc.dei.unipd.it/education.html`.

[3] M. Geronazzo, S. Spagnol e F. Avanzini, *Hearing distance: a low-cost model for near-field binaural effects*, 2012.

[4] S. Spagnol e F. Avanzini, *Real-time binaural audio rendering in the near field*, Proc. 6th Int. Conf. Sound and Music Computing (SMC09), 2009.

[5] F. Bianchi, *Inventare il suono con Pure Data, Manuale introduttivo di musica elettronica*, 2010, reperibile all'indirizzo `http://www.webprog.it/fileguida/Inventare_il_suono.pdf`.

[6] P. Zahorik, D. S. Brungart e A. W. Bronkhorst, *Auditory distance perception in humans: a summary of past and present research*, Acta Acustica united with Acustica, vol. 91, pp. 409-420, 2005.

[7] C. P. Brown e R. O. Duda, *A Structural Model for Binaural Sound Synthesis*, IEEE transaction on speech and audio processing, vol. 6, n. 5, pp. 476-488, 1998.

[8] *Eclipse*, `www.eclipse.org/`.

[9] *Pure Data*, `http://puredata.info/`.

[10] Wiki del progetto pdsmc, `http://smc.dei.unipd.it/redmine/projects/pdsmc/wiki/External_C++`.

[11] Wikipedia, `http://it.wikipedia.org/wiki/Metodo_dell'interpolazione_lineare`.

[12] J. M- Zmölnig, *HOWTO write an External for puredata*, reperibile all'indirizzo `http://pdstatic.iem.at/externals-HOWTO/pd-externals-HOWTO.pdf`.

[13] U. Zölzer, *Digital Audio Signal Processing, Second Edition*, J. Wiley & Sons, cap. 5, 2008.