



UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

**UNA PIATTAFORMA PER IL  
RENDERING AUDIO-APTICO  
DI INTERAZIONI CONTINUE**

**Relatore: Dott. Federico Avanzini**

**Laureando: Andrea Maglie**

Anno Accademico 2005/2006



*A mia mamma  
e ai miei due nonni.  
Vi voglio bene.*



# Ringraziamenti

Ringrazio i professori Federico Avanzini e Giovanni De Poli, perché mi hanno dato l'opportunità di fare esperienze completamente nuove, svolgendo ricerche su argomenti affascinanti.

Ringrazio mia madre e la mia famiglia per avermi sempre sostenuto, sia moralmente che economicamente, e per aver sempre condiviso ogni mia scelta.

Un grazie particolare a Giulia, a tutti i compagni di corso e ai ragazzi del laboratorio SMC, perché hanno saputo rendere le giornate di studio molto più leggere.

Desidero infine ringraziare tutte le persone che, molto gentilmente, si sono rese disponibili a partecipare agli esperimenti condotti durante lo svolgimento di questa tesi.



# Indice

<b>Sommario</b>	<b>xv</b>
<b>Introduzione</b>	<b>xvii</b>
<b>1 La percezione aptica</b>	<b>1</b>
1.1 Il tatto . . . . .	1
1.2 Neurofisiologia del tatto . . . . .	2
1.3 Aspetti sensoriali del tatto . . . . .	3
1.3.1 Sensibilità e risoluzione . . . . .	3
1.3.2 Effetti della posizione del corpo e dell'età . . . . .	4
1.3.3 Manipolazioni basate sulla sensibilità . . . . .	4
1.4 Percezione aptica delle proprietà degli oggetti e delle superfici . .	5
1.4.1 Ruvidità . . . . .	5
1.4.2 Peso . . . . .	6
1.4.3 Curvatura . . . . .	7
1.4.4 L'esplorazione manuale nella percezione delle proprietà degli oggetti . . . . .	7
1.5 Percezione aptica dello spazio . . . . .	8
1.5.1 Percezione di pattern bidimensionali e tridimensionali . . .	9
1.6 Memoria aptica . . . . .	10
1.7 Feedback aptico . . . . .	11
<b>2 La percezione uditiva</b>	<b>13</b>
2.1 Dall'evento all'esperienza . . . . .	14
2.1.1 Vibrazioni dei solidi . . . . .	16
2.1.2 Eventi aerodinamici . . . . .	16
2.1.3 Liquidi . . . . .	17
2.1.4 Eventi che producono suoni complessi . . . . .	17
2.2 Descrivere gli eventi . . . . .	18
2.3 Percezione e psicofisica . . . . .	19
2.4 Ascoltare e riconoscere la sorgente . . . . .	19

2.5 I modelli fisici . . . . .	20
<b>3 Modelli audio</b>	<b>23</b>
3.1 Il modello di rotolamento . . . . .	23
3.1.1 L'interazione di rotolamento con il modello di impatto come blocco di base . . . . .	23
3.1.2 La superficie . . . . .	25
3.1.3 Il modello di impatto . . . . .	27
3.1.4 Caratteristiche di alto livello . . . . .	27
3.2 Tessiture della superficie . . . . .	28
3.3 Implementazioni dei modelli in Pure Data . . . . .	31
3.3.1 Cos'è Pure Data . . . . .	31
3.3.2 Gestione dei segnali audio . . . . .	33
3.3.3 Scrivere <i>external</i> per Pure Data . . . . .	36
3.3.4 Librerie utili per Pure Data . . . . .	38
3.3.5 Implementazione della patch generatrice di rumore frattale	40
3.3.6 Implementazione della patch generatrice di rumore di sfregamento . . . . .	42
<b>4 I dispositivi aptici</b>	<b>55</b>
4.1 Cosa sono i dispositivi aptici . . . . .	55
4.2 Tipi di dispositivi . . . . .	57
4.3 Struttura di una pipeline di rendering aptico . . . . .	60
4.4 Rendering dei dettagli della superficie . . . . .	63
4.4.1 Rendering di superfici smussate . . . . .	64
4.4.2 Rendering delle tessiture aptiche . . . . .	64
4.4.3 Rendering delle superfici con frizione . . . . .	65
4.5 Generazione di tessiture casuali con il metodo stocastico . . . . .	65
4.5.1 Modelli stocastici . . . . .	66
4.5.2 Procedure implicite e sintesi spettrale di Fourier . . . . .	67
4.5.3 Filtraggio . . . . .	67
<b>5 Il Phantom® Omni™</b>	<b>69</b>
5.1 Le interfacce Phantom . . . . .	69
5.2 Il toolkit OpenHaptics™ . . . . .	71
5.3 Creazione di un ambiente aptico . . . . .	71
5.3.1 Forze dipendenti dal moto . . . . .	72
5.3.2 Forze dipendenti dal tempo . . . . .	73
5.4 Contatti e vincoli . . . . .	73
5.5 Sincronizzazione . . . . .	74

5.6	Convenzioni nell’interfaccia grafica e aptica . . . . .	75
5.7	Programmare con le HD API . . . . .	75
5.7.1	Operazioni del dispositivo . . . . .	76
5.7.2	Frame aptici . . . . .	78
5.7.3	Operazioni dello scheduler . . . . .	78
5.7.4	Stato del sistema . . . . .	79
5.8	Programmare con le HL API . . . . .	79
5.8.1	Generazione delle forze . . . . .	80
5.8.2	Threading . . . . .	80
5.8.3	Struttura della programmazione con le HL API . . . . .	80
5.8.4	Setup del dispositivo . . . . .	82
5.8.5	Frame aptici . . . . .	82
5.8.6	Rendering delle forme . . . . .	84
5.8.7	Mappatura del dispositivo aptico sulla scena . . . . .	87
5.8.8	Proprietà dei materiali ed effetti aptici . . . . .	88
5.8.9	Eventi . . . . .	89
<b>6</b>	<b>L’applicazione Phantom Friction</b>	<b>91</b>
6.1	Scopo del lavoro . . . . .	91
6.2	Implementazione grafica . . . . .	92
6.2.1	Rappresentazione della scena . . . . .	93
6.2.2	Rendering delle ombre . . . . .	94
6.2.3	Aggiunta di texture grafiche . . . . .	100
6.3	Implementazione aptica . . . . .	100
6.3.1	Rendering aptico degli oggetti virtuali . . . . .	100
6.3.2	Rilevamento dei contatti e rendering delle tessiture aptiche	102
6.4	Scambio dei dati tra l’applicazione Phantom Friction e le patch in Pure Data . . . . .	107
6.4.1	La scrittura e la lettura della memoria da parte dell’applicazione Phantom Friction . . . . .	113
6.4.2	La scrittura e la lettura della memoria da parte delle patch in Pure Data . . . . .	113
6.4.3	Analisi delle prestazioni . . . . .	115
<b>7</b>	<b>Esperimenti sulla percezione audio–aptica</b>	<b>117</b>
7.1	Introduzione . . . . .	117
7.2	Studi precedenti . . . . .	117
7.2.1	Lederman – Percezione uditiva delle tessiture (1978) . . . . .	118
7.2.2	Guest, Catmur e Lloyd – Interazioni audio–aptiche nella percezione della ruvidità (2002) . . . . .	119

7.2.3 Lederman, Klatzky, Morgan e Hamilton – Integrazione delle informazioni multimodali sulla tessitura di una superficie tramite una sonda (2002) . . . . .	121
7.3 Aspetti innovativi . . . . .	123
7.4 Attrezzature e organizzazione sperimentale . . . . .	124
7.5 Esperimento 1 . . . . .	126
7.6 Esperimento 2 . . . . .	128
7.7 Esperimento 3 . . . . .	131
7.8 Risultati sulla confidenzialità delle risposte . . . . .	133
7.9 Risultati sul questionario post–esperimento . . . . .	133
7.9.1 Esperimento 1 . . . . .	133
7.9.2 Esperimento 2 . . . . .	135
7.9.3 Esperimento 3 . . . . .	137
7.10 Conclusioni sperimentali . . . . .	138
<b>Conclusione</b>	<b>141</b>
<b>Bibliografia</b>	<b>143</b>

# Elenco delle figure

1.1	Sezione verticale della pelle della mano, con la locazione dei quattro tipi di meccanorecettori.	2
1.2	Procedure di esplorazione aptica delle proprietà degli oggetti.	7
2.1	Esempio di sorgente di onde sonore	15
2.2	Descrizione gerarchica degli eventi di tutti i giorni che producono suoni.	18
3.1	Tracciato del movimento di una palla che segue un profilo di superficie $s(x)$	24
3.2	Tracciato della curva di offset effettiva risultante dalla superficie $s(x)$ .	25
3.3	Approssimazione del tracciato compiuto dalla palla durante il rotolamento.	25
3.4	Ulteriore approssimazione del tracciato di rotolamento.	26
3.5	Spettro di ampiezza del rumore frattale	30
3.6	L'interfaccia grafica di Pure Data.	31
3.7	Semplice esempio di patch per Pure Data.	32
3.8	Linee del tempo per la computazione audio e la computazione di controllo in Pure Data	34
3.9	Utilizzo del modulo <i>simple1</i> basato su <i>flext</i>	40
3.10	La patch <i>surface_modeler</i> che implementa la generazione di rumore frattale.	41
3.11	La subpatch <i>_initialize_fractal_noise</i> .	43
3.12	La subpatch <i>_cascade</i> .	43
3.13	La patch che implementa la generazione di rumore di frizione	44
3.14	La subpatch <i>holy_roller~</i>	44
3.15	Blocchi di elaborazione del valore del diametro.	46
3.16	L'oggetto <i>_clip_velo+fade</i>	46
3.17	Gli oggetti <i>clip_exp~</i> e <i>circ_max_filter~</i> .	48
3.18	La subpatch <i>_surface_tracer~</i>	50

3.19 L'oggetto <i>impact_modalb~</i> . . . . .	51
3.20 La subpatch <i>_par_to_list4</i> . . . . .	52
3.21 La subpatch <i>_modal_object_parameters3_2</i> e i parametri del risonatore. . . . .	52
3.22 L'oggetto <i>impact_modalb~</i> e la divisione tra i gruppi di argomenti di costruzione. . . . .	53
3.23 Implementazione dell'amplificazione proporzionale alla radice quadrata del prodotto tra velocità e forza normale. . . . .	53
 4.1 Uso di un braccio meccanico per la manipolazione di molecole in un ambiente di realtà virtuale. . . . .	56
4.2 Sistema di feedback aptico usato per la pianificazione di interventi chirurgici. . . . .	57
4.3 Interazione aptica tra utente e dispositivo. . . . .	58
4.4 Dispositivo aptico Phantom® Desktop™ di SensAble Technologies.	59
4.5 Dispositivo aptico CyberForce™ . . . . .	60
4.6 I guanti (esoscheletri) aptici CyberGrasp™ e CyberGlove™ . . . . .	61
4.7 Processo associato al rendering delle forze . . . . .	61
4.8 Interazioni aptiche point-based e ray-based. . . . .	62
4.9 Tecniche di rendering aptico dei dettagli di una superficie . . . . .	63
4.10 Texture bidimensionale composta da due pdf gaussiane. . . . .	66
4.11 Schema a blocchi per modellare una texture tramite filtraggio . . . . .	68
 5.1 Il dispositivo aptico Phantom® Omni™ di SensAble Tecnologies.	70
5.2 Rappresentazione del punto SCP ( <i>surface contact point</i> ). . . . .	74
5.3 Schema di programmazione con le HD API. . . . .	77
5.4 Schema di programmazione con le HL API. . . . .	81
5.5 Collocazione dei frame con le HL API. . . . .	83
5.6 Mappatura dallo spazio di lavoro aptico alla scena grafica. . . . .	87
 6.1 Interfaccia grafica dell'applicazione <i>Phantom Friction</i> . . . . .	92
6.2 Esempio di ombre ottenute tramite trasposizione bidimensionale degli oggetti . . . . .	97
6.3 Esempio di applicazione dello shadow mapping. . . . .	99
6.4 Schema di utilizzo di un'area di memoria condivisa per lo scambio dei dati tra il processo di sintesi aptica e quello di sintesi audio. .	109
6.5 L'oggetto <i>ReadOperativo</i> e le sue connessioni. . . . .	113
6.6 Filtraggio passa-basso del valore associato al profilo della superficie frattale. . . . .	115
 7.1 Setup degli esperimenti di Guest, Catmur e Lloyd . . . . .	120

---

7.2	Andamenti delle stime della ruvidità come funzione dello spazio tra gli elementi . . . . .	122
7.3	Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al primo esperimento. . . . .	127
7.4	Grafico boxplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al primo esperimento . . . . .	127
7.5	Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al secondo esperimento (primo gruppo). . .	129
7.6	Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al secondo esperimento (primo gruppo) . . . . .	129
7.7	Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al secondo esperimento (secondo gruppo). .	130
7.8	Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al secondo esperimento (secondo gruppo). La linea orizzontale rappresenta la mediana, le linee superiori e inferiori i limiti dei quartili, la linea verticale l'estensione dei dati. . . . .	130
7.9	Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al terzo esperimento. . . . .	132
7.10	Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al terzo esperimento . . . . .	132
7.11	Risultati delle risposte al questionario post–sperimentale relativo al primo esperimento . . . . .	134
7.12	Risultati delle risposte al questionario post–sperimentale relativo al secondo esperimento . . . . .	135
7.13	Risultati delle risposte al questionario post–sperimentale relativo al secondo esperimento (primo gruppo) . . . . .	136
7.14	Risultati delle risposte al questionario post–sperimentale relativo al secondo esperimento (secondo gruppo) . . . . .	136
7.15	Risultati delle risposte al questionario post–sperimentale relativo al terzo esperimento . . . . .	137



# Sommario

Il termine *rendering* è solitamente adottato nella *Computer Graphics* per indicare il processo di generazione di un' immagine (e più in generale di un ambiente virtuale) a partire da un modello, utilizzando strumenti hardware e software. In anni recenti è entrato in uso il termine *rendering multimodale* per indicare strategie di rendering che coinvolgono altre modalità oltre a quella visiva, in particolare quella uditiva e quella aptica. In questo contesto sorgono nuovi problemi da affrontare, quali la sincronizzazione e la coerenza tra le diverse modalità, necessarie a garantire una percezione unitaria.

Con *sintesi aptica* si indica la realizzazione di un interfacciamento tra applicazione e utente in grado di trasmettere in modo bidirezionale sensazioni legate al senso del tatto, e ciò avviene tramite l'utilizzo di un dispositivo di *force feedback*. Invece di usare campioni sottoposti ad una pre-elaborazione, la *sintesi audio* è basata su modelli fisici, i quali non descrivono il suono stesso ma la sorgente e gli eventi che lo generano. Lo scenario audio-aptico simulato dipende da alcuni semplici parametri di significato fisico.

Viene affrontato il problema di realizzare un'applicazione di realtà virtuale in grado di simulare tessiture audio e aptiche di superfici, con il duplice scopo di creare un ambiente destinato ad un utilizzo sperimentale e di realizzare una base per l'implementazione di più sofisticate applicazioni, utilizzabili ad esempio per training medico, controllo remoto o intrattenimento. Resta al di fuori degli obiettivi la creazione di tessiture grafiche. La simulazione deve essere caratterizzata da latenza minima, consentendo un'esecuzione in tempo reale. Se solitamente, nella realtà virtuale, si parla di rappresentazione solo dal punto di vista grafico, in questa tesi tale aspetto passa in secondo piano, in favore delle rappresentazioni sonora e aptica.

Gli strumenti hardware comprendono un personal computer interfacciato ad un dispositivo aptico, il Phantom® Omni™ di SensAble Technologies; gli strumenti software invece sono un'interfaccia grafica tridimensionale e l'implementazione dell'algoritmo di sintesi audio.

Partendo da un modello che descrive la generazione di un rumore frattale e da altri modelli che descrivono i contatti tra un oggetto e la superficie sulla quale

---

questo si muove, si è realizzata l'implementazione parametrica dell'algoritmo che sintetizza il suono prodotto da un corpo che striscia su un altro. Un altro algoritmo è stato sviluppato per ricreare delle tessiture aptiche sugli oggetti virtuali (senza l'utilizzo di tessiture grafiche). I parametri che lo guidano sono alcuni dei parametri che guidano la sintesi sonora: avviene quindi uno scambio di dati tra i due algoritmi, i quali vengono eseguiti contemporaneamente con latenza minima, in modo che il rendering audio e aptico risultino sincronizzati. L'applicazione così realizzata è stata utilizzata per effettuare degli esperimenti psicofisici sulla percezione bimodale, i cui risultati hanno confermato la validità dei modelli e delle loro implementazioni.

La tesi è organizzata come segue: nella prima parte verranno introdotti i concetti di base sulla percezione audio e aptica; si passa poi alla descrizione dei modelli fisici implementati nella sintesi audio e del dispositivo usato per la sintesi aptica, unitamente agli strumenti di sviluppo utilizzati (*Pure Data* per la parte audio, il toolkit *OpenHaptics* per la parte aptica). Infine viene illustrato come le componenti hardware e software sono state integrate tra loro e come sono state usate per svolgere degli esperimenti di percezione bimodale.

# Introduzione

Se nel campo della computer graphics le tessiture (*texture*) sono state studiate estensivamente allo scopo di rendere sempre più reali gli ambienti virtuali, altrettanto non si può dire per quanto riguarda la *computer haptics*, dove con tale termine si intende la scienza che studia la simulazione e lo scambio tra utente e computer di informazioni legate al senso del tatto. Tuttavia, per creare un ambiente virtuale realistico, è indispensabile fornire all'utente un ritorno di suono in corrispondenza al verificarsi di un evento (come un contatto tra oggetti).

Quando tocchiamo la superficie di un oggetto reale ad occhi chiusi, raccogliamo principalmente due tipi di informazione: il primo tipo è costituito dalle informazioni percepite tramite le dita e le mani, ovvero tramite il senso del tatto; il secondo tipo è invece costituito dalle informazioni uditive, catturate dai suoni che vengono generati nel momento in cui muoviamo le nostre dita sulla superficie. La piattaforma descritta è stata realizzata proprio basandosi su questi concetti, estendendoli all'interazione con oggetti simulati.

L'obiettivo a cui mira questa tesi è la realizzazione di un sistema che permetta l'interazione visiva, aptica e sonora con un ambiente virtuale (dove la parola “aptica” è legata al senso di tocco attivo):

- *visiva* in quanto l'utente può visualizzare su uno schermo gli oggetti virtuali con cui interagisce;
- *aptica* perché gli oggetti virtuali possono essere “sentiti” al tatto;
- *audio* perché, come in un ambiente reale, sia possibile ascoltare i suoni provocati dalle interazioni.

In particolare l'attenzione viene concentrata sulla realizzazione di tessiture sulle superfici, le quali, attraverso il controllo di alcuni parametri di significato fisico, possano far sentire (sia tramite il tatto che l'udito) all'utente superfici di diverso livello di ruvidità (non ci preoccupiamo invece delle tessiture grafiche). La sintesi dell'audio non ha più come elemento centrale il segnale che viene prodotto e percepito, ma la *sorgente* del suono: attraverso modelli fisici si cerca quindi di descrivere la sorgente (o le sorgenti) sonore unitamente agli eventi che producono

---

il suono stesso. Variando un insieme ristretto di parametri è così possibile simulare diverse sorgenti; gli stessi parametri possono poi essere utilizzati per controllare la simulazione aptica.

La rappresentazione grafica dell’ambiente tridimensionale virtuale è stata fatta sfruttando le API OpenGL; si è scelto di mantenere questa componente molto semplice, infatti l’ambiente è costituito da un solo oggetto che si presenta liscio e monocromatico alla vista. Per rendere possibile la rappresentazione aptica invece è necessario eseguire un interfacciamento tra l’applicazione grafica e un dispositivo di force feedback (il Phantom® Omni™ di SensAble Technologies in questo caso); utilizzando tale dispositivo sarà possibile *toccare* ogni oggetto dell’ambiente virtuale. Infine la sintesi dell’audio è stata realizzata separatamente tramite l’ambiente di programmazione visuale *Pure Data*, implementando dei modelli fisici la cui validità è già consolidata. Le tre componenti possono essere eseguite contemporaneamente, realizzando una completa simulazione in tempo reale che può prestarsi ad una molteplicità di usi diversi.

Nel primo capitolo si descrive brevemente il senso del tatto, in particolare come il corpo umano raccoglie ed elabora le informazioni tattili; nel secondo invece viene analizzato il modo di percepire da parte dell’uomo i suoni nell’ambiente circostante.

Nel terzo capitolo si illustrano per prima cosa i modelli fisici su cui viene basata la sintesi dell’audio. I modelli comprendono la realizzazione di tessiture stocastiche a partire da un rumore filtrato e il rendering di suoni di contatto tramite una sintesi modale. Successivamente viene descritto il funzionamento dell’ambiente di programmazione Pure Data e dei moduli esterni utilizzati; segue una spiegazione dettagliata su come sono stati implementati i modelli fisici in *patch* eseguibili in tale ambiente.

Il capitolo 4 riporta una panoramica sui dispositivi aptici, mentre il capitolo successivo illustra nel dettaglio il dispositivo Phantom® Omni™, unitamente al toolkit OpenHaptics utilizzato per la programmazione di tale dispositivo.

Nel capitolo 6 si tratta la programmazione dell’interfaccia grafica e dell’interfaccia aptica; viene spiegato come avviene il rendering delle forze e come sono realizzate le tessiture aptiche. Infine si spiega come queste componenti siano state integrate tra loro e come avviene la comunicazione tra l’interfaccia grafica/aptică e l’algoritmo di sintesi audio creato ed eseguito in Pure Data.

La parte finale (capitolo 7) è dedicata agli esperimenti di percezione bimodale audio–aptică: dopo aver presentato una panoramica sugli studi che sono già stati condotti in questo campo, vengono illustrate le differenze rispetto a questi e gli aspetti innovativi degli studi svolti in questa tesi. Sono stati svolti tre esperimenti: il primo in condizione di variazione concorde degli stimoli audio e aptico, gli

---

altri in condizioni di variazione discorde dei due stimoli. In conclusione vengono riportati i risultati e le discussioni sulle percezioni avute dai partecipanti.



# Capitolo 1

## La percezione aptica

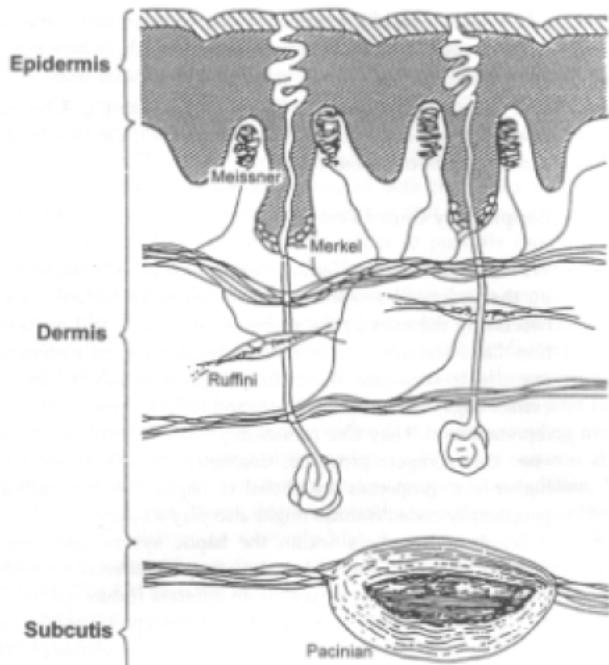
### 1.1 Il tatto

Il senso del tatto può essere visto come un sistema multisensoriale attivo. Il termine multisensoriale indica che le modalità del tatto riguardano vari sistemi sensoriali, più precisamente:

- sistema cutaneo;
- sistema cinestetico;
- sistema aptico.

Il sistema cutaneo riceve degli input sensoriali dai meccanorecettori (le terminazioni nervose che rispondono a stimolazioni meccaniche) situati nella pelle. Il sistema cinestetico (cioè quella parte del sistema nervoso che si occupa della percezione della posizione e del movimento degli arti) riceve informazioni dai meccanorecettori posti nei muscoli, nei tendini e nelle giunture. Infine, il sistema aptico utilizza le informazioni che provengono da questi due sistemi. Il termine *aptico* è associato al tocco attivo, e nella vita di tutti i giorni il tocco è proprio di questo tipo: muovendo gli arti e la pelle su superfici ed oggetti, i sensori tattili vengono stimolati, rivelando moltissime importanti proprietà del mondo che ci circonda.

Nel corso degli anni sono state formulate diverse definizioni di tocco attivo o passivo. Secondo Gibson [13] il tocco è passivo quando è assente il movimento volontario dei muscoli, mentre il tocco attivo è costituito dall'esplorazione degli oggetti tramite comandi inviati dal cervello ai muscoli. Loomis e Lederman hanno rielaborato il pensiero di Gibson per ottenere una classificazione del sistema sensoriale in base agli input usati; secondo tale classificazione esistono cinque differenti modalità di tocco [28]:



**Figura 1.1:** Sezione verticale della pelle della mano, con la locazione dei quattro tipi di meccanorecettori.

1. percezione tattile (cutanea);
2. percezione cinestetica passiva (risposta cinestetica senza movimento volontario);
3. percezione aptica passiva (risposte cinestetica e cutanea senza movimento volontario);
4. percezione cinestetica attiva;
5. percezione aptica attiva.

Solo negli ultimi due casi l'osservatore ha un controllo motorio sul processo di esplorazione tattile.

Un'ulteriore classificazione può essere fatta tra le sensazioni meccaniche del tocco (come pressione e posizione) e le sensazioni legate alla temperatura e al dolore; in tal caso non è diverso solo il tipo di sensazione, ma è diversa anche la tipologia di percezione neurale.

## 1.2 Neurofisiologia del tatto

La pelle è l'organo sensoriale più grande del nostro corpo; nell'adulto medio si estende per circa 2 metri e pesa dai 3 ai 5 chilogrammi. È composta di due strati: l'*epidermide* (la parte più esterna) e il *derma* (la parte interna); in entrambi gli

strati si trovano i meccanorecettori (figura 1.1), i responsabili della traduzione degli stimoli meccanici in stimoli neurali. Si può considerare un ulteriore strato compreso tra il derma e i muscoli: l'*ipoderma*; esso contiene tessuti connettivi e grasso sottocutaneo, oltre alle terminazioni dei meccanorecettori.

La pelle della mano contiene quattro diversi tipi di meccanorecettori, distinguibili in base all'area percettiva e alla risposta agli stimoli.

**Unità ad adattamento veloce** – Le unità ad adattamento veloce (FA - fast adapting) mostrano una rapida risposta alle deformazioni della pelle. Se la zona di ricezione degli stimoli è piccola e ben definita si parla di unità FAI, mentre le unità FAII sono costituite da zone ricettive più grandi e con confini poco definiti.

**Unità ad adattamento lento** – Le unità ad adattamento lento (SA - slow adapting) esibiscono una risposta continua alle deformazioni sostenute della pelle; le unità SAI hanno una forte sensibilità dinamica e mostrano una risposta irregolare alle stimolazioni sostenute, mentre le unità SAII esibiscono maggiore regolarità nella risposta pur essendo meno sensibili.

Altri tipi di recettori rispondono alle stimolazioni termiche; sono presenti unità di recettori che rispondono al caldo e altre che rispondono al freddo, e tutte sono localizzate nella parte più esterna della pelle. Assieme a questi si trovano anche i recettori del dolore.

I meccanorecettori presenti nei muscoli, nei tendini, nelle giunture e nella pelle delle mani contribuiscono al senso cinestetico di movimento degli arti; le terminazioni di diametro più largo codificano il tasso di cambiamento della lunghezza delle fibre muscolari e le vibrazioni; le terminazioni più piccole sono più sensibili nella fase statica dell'attività muscolare.

## 1.3 Aspetti sensoriali del tatto

### 1.3.1 Sensibilità e risoluzione

Molti esperimenti sono stati fatti per capire quali sono le soglie di reazione umane alle deformazioni meccaniche della pelle, per poi scalare l'ampiezza delle sensazioni in corrispondenza all'ampiezza degli stimoli e trovare le relazioni tra le reazioni recettive e le caratteristiche degli stimoli alle diverse soglie:

- La capacità di risoluzione spaziale della pelle è stata misurata come la minima distanza tra due punti tale che questi vengano percepiti come distinti; utilizzando una griglia di punti, la capacità di risoluzione valutata è di circa 1 millimetro.

- Gli esperimenti sulla capacità di risoluzione temporale, valutata in termini di sensibilità alle vibrazioni, hanno mostrato che gli adulti sono in grado di cogliere vibrazioni fino a 700 Hz, cioè possono distinguere intervalli temporali di circa 1.4 millisecondi. Analizzando invece la capacità di individuare come successivi due impulsi, ciascuno di un millisecondo, si è trovato che i due devono essere separati almeno di 5.5 millisecondi.

Tutto ciò dimostra che la mano ha una capacità risolutiva spaziale migliore dell'orecchio ma minore dell'occhio, mentre per la risoluzione temporale la situazione si inverte, avendosi che la mano è migliore dell'occhio ma peggiore dell'udito.

### 1.3.2 Effetti della posizione del corpo e dell'età

La sensibilità della pelle varia in base alla parte del corpo che viene stimolata: ad esempio il viso rileva forze di bassa entità, mentre le dita sono più efficienti nell'elaborare informazioni spaziali. Gli effetti dell'età sono stati studiati esaminando le soglie di rilevamento delle vibrazioni: con l'avanzare dell'età queste soglie aumentano, principalmente a causa della perdita di recettori. Lo stesso si verifica per le soglie di rilevamento della densità spaziale; dai 20 agli 80 anni si verifica un aumento di circa 1% all'anno delle soglie di discriminazione della distanza tra due punti e nel rilevamento del loro orientamento rispetto alle dita.

### 1.3.3 Manipolazioni basate sulla sensibilità

Le informazioni ricevute dalla cute giocano un ruolo fondamentale nell'interazione con gli oggetti. Basti pensare al fatto che le persone dotate di alte soglie di discriminazione tendono ad afferrare gli oggetti con più forza per poterli manipolare; per soggetti con gravi problemi la manipolazione può diventare impossibile.

Afferrare e manipolare un oggetto richiede che la presa e le forze vengano coordinate lungo una sequenza di stadi; i recettori forniscono le informazioni necessarie a dosare e coordinare le forze per compiere queste azioni. Ma le persone non usano solo le informazioni che vengono ricevute istantaneamente dal cervello: vengono sfruttate anche le conoscenze acquisite nelle esperienze passate circa il peso e le altre proprietà degli oggetti. Ciò conduce all'uso di movimenti muscolari già programmati e nell'adattamento di questi alle nuove proprietà degli oggetti manipolati.

## 1.4 Percezione aptica delle proprietà degli oggetti e delle superfici

Secondo Klatzky e Lederman [24], il sistema aptico inizia l'estrazione di informazioni già a partire dalle unità periferiche (i recettori). Ciò contrasta con il metodo di funzionamento degli organi visivi: la percezione visiva di un oggetto diventa ben definita solo in seguito a varie elaborazioni di alto livello.

Sono state fatte varie distinzioni tra le tipologie di informazioni che possono essere estratte dalla manipolazione di un oggetto o una superficie. Una prima discriminazione è tra proprietà *geometriche* e proprietà del *materiale*:

**Proprietà geometriche** – Le proprietà geometriche sono specifiche di un oggetto e possono essere divise in *dimensione* e *forma*; inoltre possiamo considerare geometrie a livello microscopico e a livello macroscopico. A livello microscopico un oggetto è abbastanza piccolo da ricoprire solo una limitata regione della pelle, come la punta di un dito; ciò produce una deformazione sulla pelle che viene codificata dai meccanorecettori (in particolare dai SAI), ricavando una mappa della disposizione dell'oggetto e le sue profondità. Nel caso macroscopico l'oggetto viene avvolto dalle mani (o dagli arti in generale), raccogliendo informazioni dai sensori cinestetici e da zone della pelle che non sono continue (come le dita); la determinazione della geometria avviene integrando tra loro tutte queste informazioni.

**Proprietà del materiale** – Le proprietà del materiale possono essere distinte, secondo Klatzky e Lederman, in *tessitura*, *rigidità*, *temperatura apparente* e *peso*. Le tessiture comprendono proprietà quali ruvidità, densità spaziale e viscosità. La rigidità non sempre corrisponde alla resistenza che oppone un oggetto (si pensi al tasto di un pianoforte: il tasto è rigido, ma nel momento in cui viene premuto non oppone una forte resistenza fino a quando non viene premuto fino in fondo); ciò significa che in questo caso entrano in gioco sia le informazioni cutanee che quelle cinestetiche.

### 1.4.1 Ruvidità

Una superficie ruvida è costituita da asperità poste sopra un substrato relativamente omogeneo. La tessitura può essere microscopica o macroscopica: si parla di micro-tessitura se le asperità sono spaziate ad intervalli dell'ordine del millesimo di millimetro; si parla di macro-tessitura se gli intervalli sono di uno o due ordini di grandezza più ampi. Con intervalli oltre i 3–4 millimetri, la superficie

non appare più come dotata di tessitura ma come una superficie liscia con delle irregolarità puntuali.

Secondo vari studi, la ruvidità percepita aumenta all'aumentare dello spazio tra le asperità; l'aumento della grandezza delle asperità invece contribuisce a far avvertire la superficie come meno ruvida. La percezione è influenzata anche dalle modalità con le quali questa avviene: l'applicazione di una pressione sulla superficie con le dita porta ad aumentare la ruvidità percepita; anche la diversa velocità di esplorazione porta a considerare diversi livelli di ruvidità. Ciò che invece non influisce è se il controllo è attivo o passivo; quindi le informazioni cinestetiche giocano un ruolo marginale. Secondo il modello formulato da Taylor e Lederman, la percezione della ruvidità è basata sulla complessiva deformazione che lo stimolo provoca sulla pelle; inoltre la deformazione può essere assunta come unidimensionale.

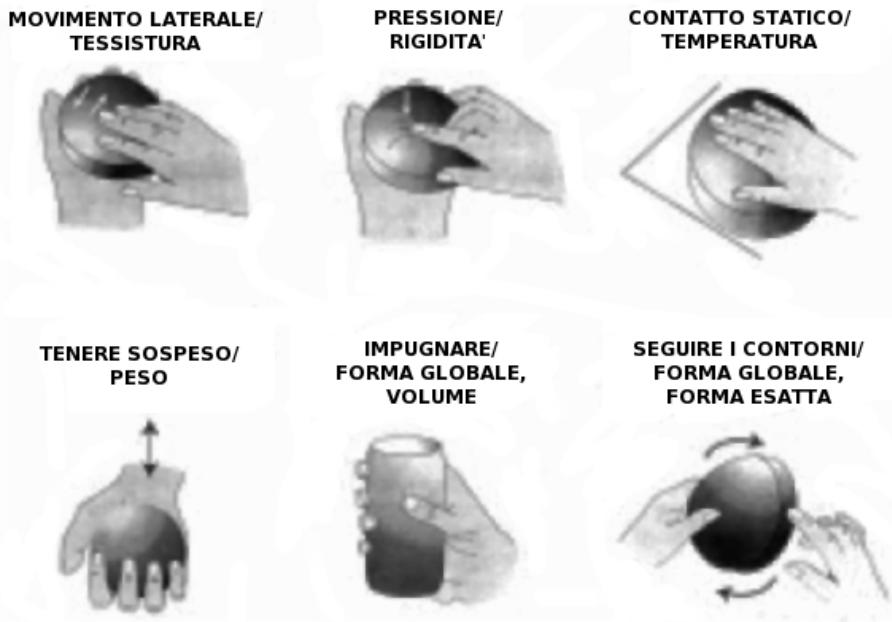
Quando gli stimoli vengono presentati sulla pelle nuda, l'uomo tende a non considerare le vibrazioni nella valutazione delle macro-tessiture (quindi lo spazio tra le asperità e la velocità di esplorazione influiscono minimamente sul giudizio); il contrario avviene nella percezione delle micro-tessiture, per le quali le vibrazioni permettono di discriminare tessiture con asperità comprese tra 0.6 e 1.6 micron spaziate di circa 100 micron.

### 1.4.2 Peso

Weber nel 1834 notò che un oggetto viene percepito come più pesante quando viene impugnato rispetto a quando viene semplicemente appoggiato sulla pelle, suggerendo che il peso percepito non dipende solamente dal suo valore oggettivo. Successivamente Chapentier e Dresslar [4] capirono che altri fattori intervengono nella percezione del peso, dato che, tra due oggetti dello stesso peso e diverse dimensioni, quello più piccolo sembra più pesante. Tali osservazioni sono state confermate dai risultati di alcuni studi, secondo i quali la percezione del peso dipende dalla resistenza che questo oppone alla rotazione (in particolare dipende dagli autovalori della matrice di rotazione).

Un altro fattore che influenza questo tipo di percezione è il materiale di cui è costituito l'oggetto: oggetti composti da materiali più densi vengono avvertiti come più pesanti; oggetti più scivolosi necessitano di una presa più forte per essere manipolati, e ciò può condurre ad avvertirli come più pesanti. L'influenza del materiale non è presente tuttavia quando si considerano oggetti di grande massa o quando oggetti di piccola massa vengono impugnati con forza.

In tutte queste percezioni è sempre presente una componente cognitiva, che



**Figura 1.2:** Procedure di esplorazione aptica delle proprietà degli oggetti.

porta l'uomo ad utilizzare informazioni già acquisite in passato per questo tipo di valutazioni.

### 1.4.3 Curvatura

Con il termine *curvatura* si intende il tasso di cambiamento dell'angolo della tangente ad una curva al variare del punto per il quale passa la tangente. La percezione della curvatura viene influenzata da eventuali altre superfici toccate in precedenza, oppure dal fatto che la curvatura sia orientata lungo le dita, che tocchi il palmo o il dorso della mano.

Durante l'esplorazione di una superficie curva, l'informazione più importante (quella che viene valutata di più nella determinazione del livello di curvatura o nella distinzione tra oggetti curvi e oggetti piani) è data dalla differenza nelle posizioni relative delle dita.

### 1.4.4 L'esplorazione manuale nella percezione delle proprietà degli oggetti

Lederman e Klatzky [23] hanno individuato l'esistenza di movimenti particolari che vengono eseguiti dalle persone nell'esplorazione delle proprietà di un oggetto, e ogni tipo di proprietà è associata ad una diversa *procedura di esplorazione* (figura 1.2):

- il *movimento laterale* è associato all'esplorazione delle tessiture;

- con il *contatto statico* si cerca di massimizzare la zona di contatto con la superficie per individuarne la temperatura;
- *impugnare* un oggetto serve per capire a grandi linee la sua forma e il suo volume;
- esercitare una *pressione* fornisce informazioni sulla resistenza di un oggetto;
- *tenere in mano l'oggetto* dà informazioni circa il suo peso;
- l'*esplorazione dei contorni* serve a individuare i contorni precisi (e quindi la forma precisa) dell'oggetto in esame.

## 1.5 Percezione aptica dello spazio

Non esiste ancora una definizione universalmente accettata di *spazio aptico*; Lederman, Klatzky, Collins e Wardell [25] hanno introdotto una distinzione tra lo spazio che viene raggiunto ed esplorato dalle mani e lo spazio che viene esplorato tramite movimenti del corpo. Il primo tipo di spazio si può definire *spazio manipulatorio*.

La percezione aptica dello spazio è anisotropica, in quanto non è possibile applicare una metrica alle percezioni di questo tipo: esse risultano distorte rispetto alla realtà e non sono uniformi lungo lo spazio esplorato. Un primo tipo di distorsione è costituito dall'illusione verticale–orizzontale: la lunghezza di linee verticali viene sovrastimata rispetto alle stesse poste in orizzontale. Tale illusione è stata riscontrata sia in soggetti ciechi che in soggetti senza problemi di vista (quindi non è dovuta a illusioni visive) ed è fortemente influenzata dal movimento delle braccia usato durante l'esplorazione (dipende quindi dalla distanza relativa tra soggetto e oggetto esaminato).

Un altro tipo di illusione riguarda il movimento radiale o tangenziale: i movimenti radiali (verso il corpo e lontano dal corpo) tendono a dare un giudizio sovrastimato rispetto ai movimenti tangenziali di uguale estensione. Anche in questo caso è forte l'influenza della posizione degli arti; ad esempio una distanza percepita è più grande se la mano si trova vicina al corpo.

Il terzo tipo di illusione riguarda l'orientamento obliquo: riprodurre l'orientamento di un'asta è più difficile quando questa è obliqua (a 45 gradi ad esempio). Si è visto inoltre che l'effetto è maggiore quando è presente la forza gravitazionale rispetto a quando questa viene annullata con dei pesi che la controbilanciano; infatti, se l'asta è posta sul piano orizzontale, l'effetto non si presenta.

### 1.5.1 Percezione di pattern bidimensionali e tridimensionali

Di seguito sono riportati i risultati di vari studi eseguiti su diversi tipi di pattern aptici.

**Pattern vibrotattili** – Un pattern vibrotattile viene generato stimolando una parte del corpo (usualmente le dita) tramite un insieme di punti di contatto. I punti di contatto sono costituiti da una matrice di spilli che vibrano circa 230 volte al secondo; le righe sono distanti tra loro circa un millimetro mentre le colonne circa 2.5 millimetri. Due pattern che vengono presentati ad una certa distanza temporale tra loro possono sommarsi formando un nuovo pattern o produrre due distinte risposte; tali effetti si verificano anche se i pattern vengono presentati allo stesso istante su dita diverse della mano. Le interazioni spaziali si verificano se il pattern stimola aree comuni della pelle o quando provocano uno stesso tipo di reazione su zone diverse della pelle; l'abilità di discriminare i pattern è inversamente proporzionale all'area di pelle che viene stimolata da tutti i pattern.

**Pattern bidimensionali e forme libere** – I pattern bidimensionali sono composti da un insieme di sporgenze lineari o puntuali, come ad esempio il Braille. I meccanorecettori coinvolti principalmente nella percezione di queste trame sono i SAI, i quali agiscono su piccole aree e quindi sono più sensibili alle discontinuità di una superficie, e insieme producono una risposta che preserva la forma della superficie stessa. Se i pattern bidimensionali rappresentano oggetti reali, il riconoscimento di questi è guidato in gran parte dalle conoscenze visuali che si posseggono dell'oggetto in esame.

**Oggetti tridimensionali** – Contrariamente alle riproduzioni bidimensionali, gli oggetti reali vengono riconosciuti molto bene al tatto. Una causa di ciò è che nella riproduzione bidimensionale viene eliminata una dimensione, la profondità, la quale può essere ricostruita tramite la vista ma non tramite il tocco. L'esplorazione solitamente inizia con l'individuazione delle caratteristiche locali, per poi passare all'estrazione delle caratteristiche globali dell'oggetto. Oggetti simili nella forma ma con diverse caratteristiche locali vengono tendenzialmente giudicati come diversi se l'esplorazione è aptica, mentre con l'esplorazione visiva vengono giudicati simili; il risultato dell'esplorazione aptica si avvicina a quello dell'esplorazione visuale man mano che il tempo di analisi dell'oggetto aumenta. Inoltre sembra che le persone tendano ad esaminare la parte frontale di un oggetto se viene usata la vista,

mentre in un’analisi aptica si considera maggiormente la parte posteriore (quella che più spesso viene esplorata con le dita).

Sono stati eseguiti degli studi anche sull’interazione tra percezione aptica e percezione visuale, allo scopo di verificare se l’attenzione del soggetto viene rivolta automaticamente verso una certa zona dello spazio o piuttosto viene attratta da degli stimoli. Se vengono stimolati entrambi i sensi (tatto e vista) concordemente, allora l’attenzione può essere rivolta spontaneamente sia nella modalità visiva che nella modalità aptica; al contrario, una stimolazione visiva incongruente con quella aptica può condurre ad una direzione errata dell’attenzione aptica. Tale effetto è dovuto alla dominazione del senso della vista sul senso del tatto e dipende dall’età del soggetto (all’aumentare dell’età aumenta il ruolo del tatto) e dalla sua esperienza.

## 1.6 Memoria aptica

Nello studio della memoria umana è sempre stata data molta importanza agli effetti degli stimoli visivi e uditivi; d’altro canto lo studio della memoria aptica viene reso più difficile dal fatto che gli stimoli aptici possono essere facilmente modulati da quelli visivi (oppure da una memoria visiva).

Secondo Millar [32], esiste nell’uomo una memoria aptica che è a breve termine e si limita a ricordare due o tre oggetti. Quando vengono raccolte informazioni tramite il tatto, la loro rappresentazione in memoria può essere intrinseca a questa modalità o più generica. Ad esempio nell’esplorazione di piccoli pattern come il Braille o altre tessiture, la rappresentazione avviene in termini di proprietà aptiche; se invece i pattern possono essere organizzati secondo strutture spaziali, le informazioni su queste strutture vengono memorizzate assieme alle informazioni puramente aptiche. La rappresentazione aptica è intramodale anche nel senso che tali informazioni possono essere usate non solo dal tatto ma anche dalla vista; infatti spesso i pattern esplorati apticamente vengono poi riconosciuti alla vista.

I bambini riescono a discriminare quasi sempre gli oggetti che vengono loro presentati sia utilizzando sempre una stessa modalità di analisi (solo aptica o solo visiva), sia utilizzando modalità diverse (a volte aptica, a volte visiva). L’accuratezza di questi riconoscimenti tuttavia decrementa se vengono analizzati oggetti non conosciuti; si pensa che questo dipenda dal fatto che i soggetti effettuano una *categorizzazione aptica* degli oggetti in base alle loro proprietà e alla disponibilità o meno di una percezione visiva degli stessi.

La memoria umana può essere *esplicita* o *implicita*: la differenza tra le due è che nell’uso della memoria esplicita il soggetto volontariamente ricerca informa-

zioni tra i suoi ricordi. La memoria aptica può essere sia implicita che esplicita, ma con modalità diverse rispetto alla memoria visiva.

## 1.7 Feedback aptico

Se si pensa al principio di azione–reazione di Newton, si capisce come in natura non esiste il concetto di *feedback aptico*. Le variabili intensive (come la forza) descrivono in modo astratto le interazioni, mentre le variabili estensive descrivono lo stato del fenomeno osservato, e non è possibile eseguire una separazione di questi due tipi di variabili; le forze sono individuabili attraverso i loro effetti sullo stato del fenomeno e tali effetti non sono necessariamente gli stessi per fenomeni sotto la stessa influenza.

Non si può parlare quindi di feedback aptico nell’interazione tra oggetti fisici, ma solo nell’interazione uomo–uomo o uomo–macchina. Sia l’uomo che le macchine infatti possono essere visti come formati da sensori e attuatori (gli oggetti invece costituiscono entità non separabili da questo punto di vista). Come abbiamo visto, nell’uomo esistono vari tipi di recettori che raccolgono le informazioni e le trasmettono al cervello; per quanto riguarda le macchine invece è evidente che, per poter ricevere e trasmettere una forza, devono essere equipaggiate con sensori e attuatori artificiali. Come vedremo nel capitolo 4, i dispositivi e le interfacce aptiche posseggono queste caratteristiche.



# Capitolo 2

## La percezione uditiva

I suoni che percepiamo ogni giorno nascono da interazioni tra oggetti (un martello che colpisce un metallo, una moneta che cade) o da cambiamenti nelle proprietà di un singolo oggetto (come un palloncino che scoppia). Ma siamo in grado di riconoscere tali eventi fisici e le loro proprietà solo sulla base del suono prodotto? Per rispondere a questa domanda sono stati svolti numerosi studi, utilizzando diversi approcci. Uno di questi consiste nel dividere l'oggetto di studio in tre livelli:

- livello fisico;
- livello acustico;
- livello percettivo.

L'analisi delle relazioni tra il livello percettivo e il livello fisico permette di capire se le caratteristiche di un evento vengono riconosciute propriamente o vengono scalate; l'analisi delle relazioni tra livello fisico e livello acustico ci dice come variano le proprietà del segnale sonoro in base alle proprietà fisiche dell'evento; infine dall'analisi delle relazioni tra livello acustico e percettivo si capisce se e come i segnali acustici influenzano il riconoscimento e la classificazione delle proprietà dell'evento in esame.

Le caratteristiche del segnale audio possono essere raggruppate in due categorie:

- caratteristiche studiate dalla ricerca classica sulla percezione sonora, come ampiezza, durata, tonalità, timbro e attacco;
- caratteristiche della sorgente sonora.

Durante l'ascolto la nostra attenzione viene rivolta a l'una o l'altra classe di proprietà. In base a ciò è possibile dire che le persone assumono due tipi di comportamenti durante l'ascolto dei suoni [12]:

- l'ascolto di tipo *musicale* porta a riconoscere i suoni assieme alle loro proprietà;
- l'ascolto *di tutti i giorni* porta a riconoscere gli eventi e le sorgenti sonore piuttosto che le proprietà del suono.

La maggior parte della nostra esperienza nell'ascolto degli eventi può essere classificata come ascolto di tutti i giorni: ascoltiamo ciò che avviene attorno a noi, imparando cosa è importante evitare e cosa invece può offrirci una possibilità di interazione. Le dimensioni percettive e gli attributi considerati sono quelli dell'evento che produce il suono, e non quelli del suono di per sé; tale esperienza è molto diversa dall'ascolto di tipo musicale, e non può essere studiata pienamente utilizzando gli approcci tradizionali all'acustica. E' anche vero che i suoni musicali non sono rappresentativi della classe di suoni che regolarmente sentiamo:

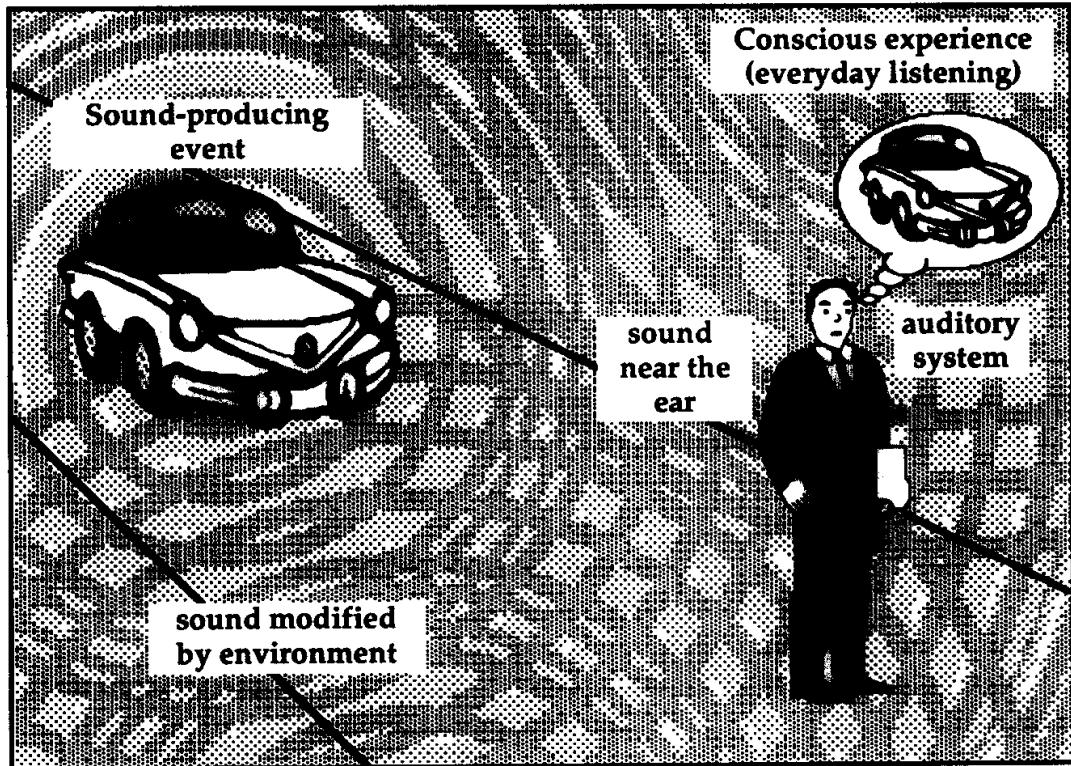
- i suoni musicali sono armonici, hanno un'evoluzione temporale semplice, non rivelano molte informazioni riguardo alla loro sorgente e variano lungo dimensioni come tonalità e ampiezza;
- i suoni di tutti i giorni sono inarmonici o rumorosi, hanno un'evoluzione temporale complessa, spesso rivelano molte informazioni riguardo alla loro sorgente e variano lungo molte dimensioni.

Per studiare quest'ultima tipologia di suoni è necessario espandere la psicoacustica in due modi: considerando le dimensioni del suono e della sua sorgente e trattando alcune variabili complesse come elementari. Tali assunzioni guidano lo sviluppo dell'*approccio ecologico* alla percezione uditiva.

In questo tipo di approccio gli stimoli elementari non necessariamente corrispondono a dimensioni fisiche altrettanto elementari, ma in alcuni casi sono costituite da eventi complessi; per questo, secondo l'approccio ecologico, lo studio della percezione deve essere rivolto a scoprire le dimensioni rilevanti per la percezione e le informazioni relative a queste.

## 2.1 Dall'evento all'esperienza

Immaginiamo di sentire l'avvicinarsi di un'automobile. Possiamo considerare tale evento come una propagazione continua di energia dalla sorgente al soggetto; lungo il percorso che questo flusso di energia compie si trovano vari ostacoli, ognuno con le sue caratteristiche ed ognuno influenzante la propagazione (figura 2.1).



**Figura 2.1:** L'esempio dell'automobile come sorgente di onde sonore; alcune onde raggiungono l'orecchio umano immutate, altre invece vengono modificate dall'ambiente.

Nel caso in esame, la prima fonte di informazione è l'automobile (la sorgente): il suono dipende da svariati fattori, come il movimento dei cilindri del motore, lo sfregamento degli ingranaggi e le vibrazioni della carrozzeria. Vengono così determinate le onde di pressione che si propagano radialmente dalla sorgente (contrariamente alla luce radiante, la propagazione radiale del suono ha una struttura ricca e fornisce molte informazioni riguardo la sua sorgente).

Successivamente il suono viene modificato dagli ostacoli che incontra nell'ambiente circostante; in particolare il suono perde energia man mano che si allontana dalla sorgente, specialmente alle alte frequenze (anche se non sono presenti ostacoli), e ciò fornisce un'informazione riguardo alla localizzazione della sorgente; se la sorgente si muove si avverte un cambio di frequenze (effetto Doppler) e un cambiamento nell'ampiezza del suono indica un'allontanamento o avvicinamento della sorgente stessa. Dato che il sistema uditivo umano è mobile, possiamo girare la testa al fine di cogliere i cambiamenti nei pattern e migliorare la localizzazione. Vediamo così come un suono dia informazioni circa un'*interazione tra materiali* in un certo *luogo* e in un determinato *ambiente*.

Ulteriori studi [12] hanno portato alla distinzione dei suoni di tutti i giorni in tre grandi categorie: solidi, liquidi e aerodinamici, in quanto raramente queste

classi vengono confuse tra di loro. Ogni classe viene poi suddivisa in base al tipo di interazione tra i materiali: ad esempio i suoni generati da solidi vibranti sono divisi in suoni di rotolamento, di sfregamento, di impatto e di deformazione. Queste classi costituiscono gli eventi base che producono dei suoni.

### 2.1.1 Vibrazioni dei solidi

Questa classe comprende eventi come la rottura di un vetro, una porta che sbatte o un'automobile in moto. Gli oggetti vibrano quando su di essi una forza viene impressa e rilasciata, portando il sistema fuori dal suo stato di equilibrio; tale forza deforma l'oggetto dalla sua configurazione originale, mentre la forza che l'oggetto oppone alla deformazione viene trasformata in energia potenziale nella nuova configurazione. Quando la forza smette di agire, l'oggetto cerca di tornare nella posizione di riposo, l'energia potenziale si trasforma in energia cinetica e l'oggetto vibra. Le vibrazioni continuano fino a quando tutta l'energia accumulata viene persa e l'oggetto torna nella posizione iniziale o trova un nuovo equilibrio. Il tipo di interazione (un urto, uno sfregamento o un rotolamento) determina sia la variazione nel tempo dell'ampiezza che lo spettro della vibrazione; la forza invece determina l'ampiezza complessiva della vibrazione. Il pattern di vibrazione è determinato anche dal tipo di materiale di cui è costituito l'oggetto, quindi dalla sua rigidità. La dimensione determina la più bassa frequenza di vibrazione, mentre la forma determina frequenza e pattern spettrale prodotto.

Tutti i parametri considerati possono essere raggruppati in due domini: il dominio della frequenza e il dominio temporale. Dal momento che la frequenza è il reciproco del tempo, è difficile separare questi due domini dal punto di vista fisico; tuttavia sono separabili dal punto di vista psicologico: i parametri nel dominio della frequenza influenzano le vibrazioni dell'oggetto, mentre i parametri nel dominio del tempo provocano effetti che diventano evidenti solo dopo alcuni cicli di vibrazioni. Gli attributi dell'oggetto (densità, dimensioni) hanno effetti sul suono nel dominio della frequenza, mentre gli attributi dell'interazione (tipo e forza) influenzano i parametri nel dominio temporale.

### 2.1.2 Eventi aerodinamici

I suoni aerodinamici vengono prodotti quando una sorgente modifica l'attuale pressione atmosferica circostante (come quando esplode un palloncino). La variazione di pressione si propaga come un'onda, la quale, se raggiunge l'orecchio e possiede particolari proprietà, può essere avvertita come suono. La maggior parte delle informazioni viene data dalla banda di frequenza del suono, dipendente dalla forza e dalla quantità della variazione di pressione. Le componenti

in alta frequenza indicano la velocità nel cambiamento di pressione, mentre le componenti in bassa frequenza dipendono dal gas coinvolto.

Un altro tipo di evento aerodinamico si verifica quando un cambiamento nella pressione imprime energia ad un oggetto provocando una sua vibrazione.

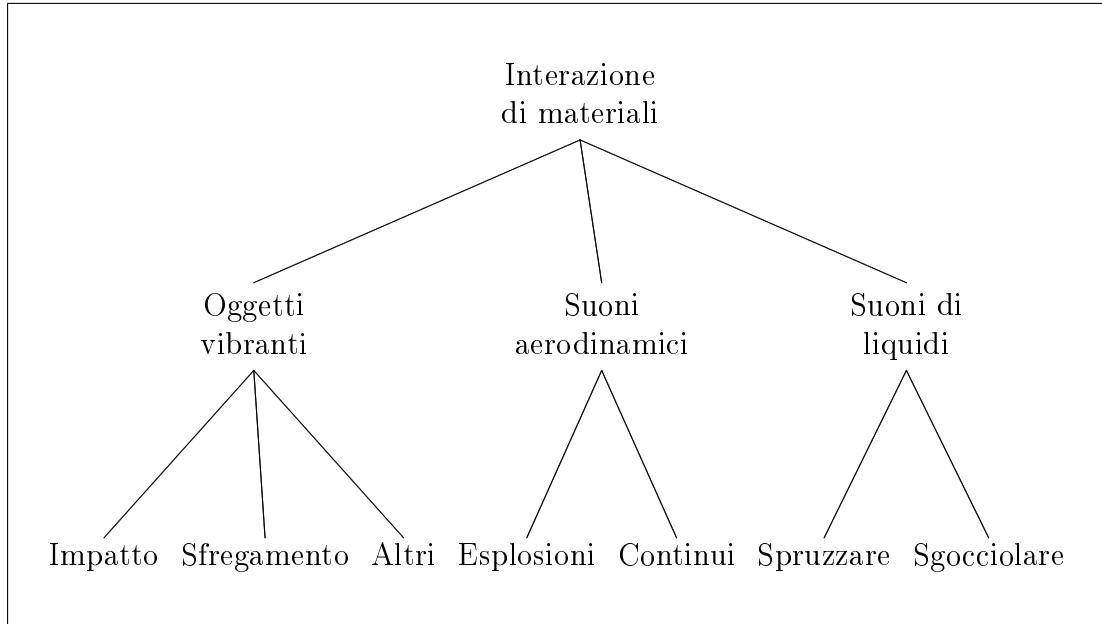
### 2.1.3 Liquidi

Gli eventi che coinvolgono liquidi dipendono da una deformazione iniziale come nella vibrazione dei solidi, ma la vibrazione non influisce sull'aria circostante in modo da provocare un suono; il suono invece è il risultato della formazione e variazione di cavità risonanti nella superficie del liquido. Per rendersene conto, basta pensare ad un piccolo oggetto che cade in un bicchiere d'acqua: al momento del contatto il liquido viene spostato dall'oggetto, formando una cavità che risuona ad una frequenza caratteristica, amplificando e modificando l'onda di pressione creata dall'impatto. Successivamente la pressione del liquido lo porta a chiudere la cavità, immergendo completamente l'oggetto. Un tale suono è quindi caratterizzato da un breve impulso seguito da altri brevi impulsi di frequenza più alta. I dettagli del suono sono determinati da massa, dimensione, velocità dell'oggetto e dalla viscosità del liquido.

### 2.1.4 Eventi che producono suoni complessi

Molti suoni dipendono da pattern complessi degli eventi descritti, o una combinazione di questi. Anche se la fisica non descrive tali eventi complessi, esistono attributi di alto livello che producono importanti effetti sul loro suono. Un esempio di attributo è l'intervallo tra eventi successivi: una sequenza di passi possono essere avvertiti come una camminata se gli intervalli tra un passo e l'altro cadono all'interno di un certo intervallo; altro esempio è la presenza di vincoli tra gli oggetti coinvolti nell'evento (sarebbe strano sentire il cigolio di una porta che si chiude lentamente accompagnato dal forte suono di una porta che viene chiusa con forza).

In tutti questi casi le sorgenti sonore possono essere considerate annidate (si pensi al suono di un'automobile, del suo motore e dei cilindri); in tal caso, un evento base può essere definito come un evento composto da una singola interazione e un singolo oggetto che produce il suono. Gli eventi complessi possono essere considerati allora combinazioni di eventi base, nelle quali la struttura della combinazione aggiunge informazioni importanti a quelle già fornite dagli eventi base. Infine, è proprio questa struttura più complessa che permette di estrarre più facilmente informazioni.



**Figura 2.2:** Descrizione gerarchica degli eventi di tutti i giorni che producono suoni.

In base alla struttura delle combinazioni possiamo distinguere tre tipi di eventi complessi:

- gli eventi definiti da un *pattern temporale* di eventi base (come il rimbalzo di una palla è composto da un pattern di impatti);
- gli eventi *residui*, dati dalla sovrapposizione di diversi eventi base;
- gli eventi *ibridi*, dati dall’interazione tra diversi tipi di materiale.

Ognuno di questi eventi complessi potenzialmente produce lo stesso profilo di sorgente sonora, più eventualmente altre proprietà specifiche: ad esempio una serie di impatti spaziati opportunamente nel tempo può portare ad individuare il rimbalzo di una palla, dando anche informazioni sul materiale e sulla simmetria della palla stessa.

Il suono prodotto da questi eventi varia in base a molte dimensioni fisiche di base; tuttavia spesso non percepiamo la variazione del suono in ogni singola dimensione, bensì avvertiamo una variazione lungo una nuova dimensione (fittizia) che incorpora tutte le altre.

## 2.2 Descrivere gli eventi

Mentre i suoni vengono descritti in base ad attributi come frequenza, ampiezza e durata, più difficile è trovare degli attributi che permettano di descrivere gli eventi di tutti i giorni.

Una possibilità è quella di classificare questi eventi in base al contesto nel quale si verificano. Ciò può essere utile se si vuole trovare un suono all'interno di una classificazione di tale tipo, ma non permette una descrizione efficiente di ciò che sentiamo, in quanto le classi non sono mutuamente esclusive (uno stesso evento può verificarsi in contesti diversi). Più interessante sembra essere una descrizione di tipo gerarchico: in questo modo gli eventi che si trovano ad un livello superiore nella gerarchia danno informazioni utili sul tipo di eventi di livello subordinato, mentre dimensioni e caratteristiche servono a descrivere le differenze tra i membri di una stessa categoria.

In base a quanto detto possiamo quindi classificare i suoni di tutti i giorni come riportato in figura 2.2: al più alto livello tutti gli eventi sono visti come interazione di materiali; al livello successivo invece vengono divisi in vibrazioni di solidi, aerodinamici e liquidi; al terzo livello la distinzione è tra eventi base.

## 2.3 Percezione e psicofisica

Come per la percezione aptica, anche la percezione sonora è un processo che si articola in tre livelli: livello fisico, neurale e mentale. In più, nel riconoscimento dei suoni le persone cercano di trarre vantaggio dall'esperienza acquisita nel passato (anche se non è possibile effettuare una stima di tale esperienza).

Esiste una differenza, nella nostra percezione uditiva, tra *ciò che è* e *ciò che sentiamo*:

- siamo in grado di sentire suoni che non esistono (la fondamentale mancante);
- non riusciamo a sentire suoni che esistono (mascheramento);
- sentiamo due diversi eventi con lo stesso insieme di stimoli (ritmi reversibili);
- possiamo sentire suoni che sono prodotti da sorgenti inesistenti nell'ambiente (musica elettronica o qualsiasi manipolazione spettrale di suoni reali).

L'uomo agisce in base a ciò che sente, non in base a ciò che esiste.

## 2.4 Ascoltare e riconoscere la sorgente

Dire che l'uomo “sente” la sorgente del suono può essere corretto se si considera un ambiente naturale: se ad esempio ci troviamo vicino ad un violinista che suona, possiamo dire che sentiamo il violino suonare. Ma ascoltando il suono di un violino proveniente da un impianto hi-fi non ci verrebbe mai in mente di dire che stiamo sentendo il cono dell'altoparlante (anche se ciò è vero); piuttosto continuiamo a

sostenere che stiamo sentendo un violino. E' opportuno quindi dire che l'uomo non sente la sorgente del suono, bensì l'uomo "rappresenta" la sorgente.

Il riconoscimento del suono e della sua sorgente è però un concetto diverso rispetto alla percezione:

1. si può avere percezione senza riconoscimento;
2. il riconoscimento può avvenire a vari livelli: è possibile riconoscere un rumore ma non una tonalità, il rombo di un'auto ma non quello di una motocicletta;
3. si possono verificare riconoscimenti fasulli: più precisamente un riconoscimento è sempre "vero" all'inizio, ma può diventare "falso" dopo aver percepito ulteriori informazioni sull'evento (come quando si pensa di sentire la pioggia e invece si tratta del rumore delle foglie mosse dal vento);
4. ci sono riconoscimenti ambigui (ad esempio quando non siamo in grado di identificare una voce come maschile o femminile);
5. lo stesso stimolo acustico può dare vita a diversi riconoscimenti a seconda che si verifichi da solo o che si ripeti (un singolo colpo di pistola viene identificato come tale, mentre una sequenza veloce di tali colpi fa pensare ad una mitragliatrice);
6. a volte la sorgente che viene riconosciuta è immateriale (si pensi all'accelerazione o decelerazione di un ritmo);
7. alcuni riconoscimenti avvengono in tempo reale, mentre altri possono avvenire "in ritardo", riguardando suoni già sentiti e memorizzati nella nostra mente.

## 2.5 I modelli fisici

I modelli fisici hanno lo scopo di sintetizzare, tramite un modello matematico, le proprietà degli oggetti reali; in particolare tramite i modelli audio si cerca di riprodurre i suoni associati a determinati eventi.

I modelli sono però sempre delle idealizzazioni, e quindi non possono rappresentare fedelmente i fenomeni che si verificano in natura. Inoltre gli stimoli provenienti da sorgenti reali variano lungo molte dimensioni, ma non tutte possono essere sintetizzate: è possibile ad esempio riprodurre le variazioni di ampiezza, timbro, durata, e dinamica; ma diventa molto difficile sintetizzare proprietà come presenza, brillantezza e contenuto espressivo. Inoltre, nella fisica reale sono

presenti molti oggetti che vibrano simultaneamente, mentre l'orecchio riceve una unica onda sonora; è necessario così estrarre da quest'ultima i segnali o le caratteristiche che possano ricreare la molteplicità di oggetti, la loro disposizione, il percorso dell'onda sonora ed eventuali ostacoli lungo il suo cammino.



# Capitolo 3

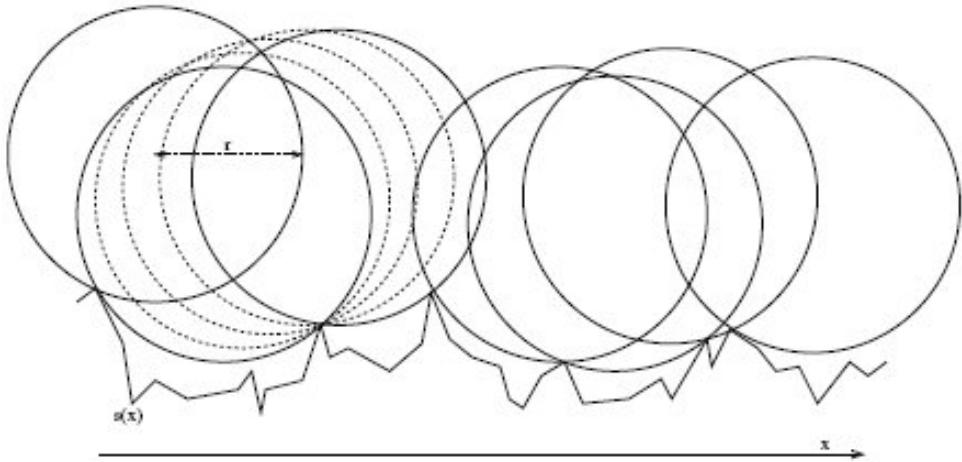
## Modelli audio

### 3.1 Il modello di rotolamento

Tra le comuni interazioni meccaniche che coinvolgono oggetti solidi, il rotolamento forma una categoria interessante anche dal punto di vista dell'audio: l'esperienza di tutti i giorni ci dice che il suono prodotto da un oggetto rotolante viene spesso riconosciuto come tale, e in generale è distinto da altri suoni come quelli dovuti allo sfregamento anche degli stessi oggetti. Ciò potrebbe essere dovuto alla natura del rotolamento come un processo di interazione continua, dove la forza mutua sugli oggetti coinvolti è descritta come un impatto senza l'aggiunta di forze di frizione perpendicolari. Oltre ad essere caratteristici, i suoni di rotolamento portano importanti informazioni: in aggiunta alle caratteristiche di risonanza degli oggetti coinvolti (che dipendono da forma, dimensione e materiale), altri attributi vengono espressi nel suono, attributi *di trasformazione*, come velocità, gravità o accelerazione/decelerazione. Lo sviluppo di un modello di rotolamento espressivo e in tempo reale da presupposti fisici, acustici e implementativi è descritto di seguito.

#### 3.1.1 L'interazione di rotolamento con il modello di impatto come blocco di base

Contrariamente ad azioni quali lo sfregare o il grattare, la forza di interazione dei due oggetti coinvolta in un semplice scenario di rotolamento (l'oggetto rotolante e il piano) è perpendicolare alla superficie di contatto (la curva media macroscopica), diretta lungo la linea che connette il punto di contatto e il centro di gravità dell'oggetto rotolante. Le condizioni di contatto devono essere modificate per riflettere le varie distanze della superficie di contatto (figura 3.1). L'oggetto rotolante è assunto come localmente sferico, senza dettagli macroscopici sulla superficie. E' possibile fare queste assunzioni dal momento che i dettagli



**Figura 3.1:** Tracciato del movimento di una palla che segue un profilo di superficie  $s(x)$ . Non si tratta del moto reale ma di una idealizzazione utile per ricavare la curva usata dal modello di impatto.

microscopici della superficie dell'oggetto rotolante possono essere semplicemente aggiunti alla superficie sulla quale l'oggetto rotola, e può essere variato il raggio di curvatura della superficie stessa; vedremo che anche l'assumere un raggio costante può essere soddisfacente per la maggior parte degli scopi. E' importante notare che il contatto tra i due oggetti durante il rotolamento è ristretto a punti distinti: il piano non viene seguito nella sua interezza.

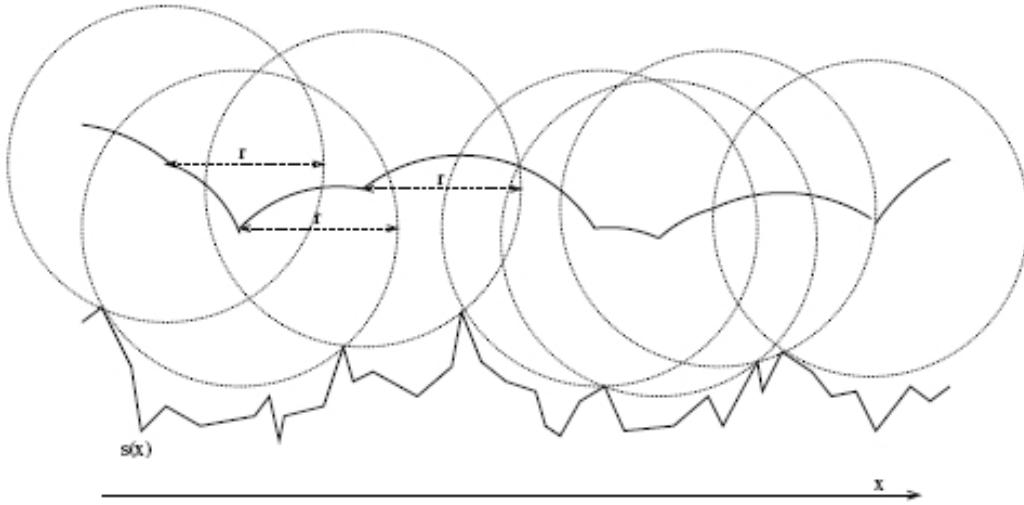
Il movimento reale dell'oggetto rotolante si differenzia da questa idealizzazione a causa dell'elasticità e dell'inerzia. In buona approssimazione, il movimento verticale del centro della palla è calcolato con un modello di impatto unidimensionale con la curva in figura 3.1. I punti di contatto e la traiettoria risultante, che idealmente dovrebbe essere applicata al modello di impatto unidimensionale, sono rappresentati in figura 3.2. Il calcolo esatto dei punti di contatto è dispendioso in termini di risorse computazionali: in ogni punto  $x$  lungo la curva della superficie, cioè per ogni punto di campionamento nel caso discreto (dove la frequenza del campionamento è la stessa del campionamento audio), deve essere calcolata la seguente funzione che descrive l'attuale punto  $p_x$ :

$$f_x(p_x) \stackrel{!}{=} \max_{q \in [x-r, x+r]} f_x(q) \quad , \quad (3.1)$$

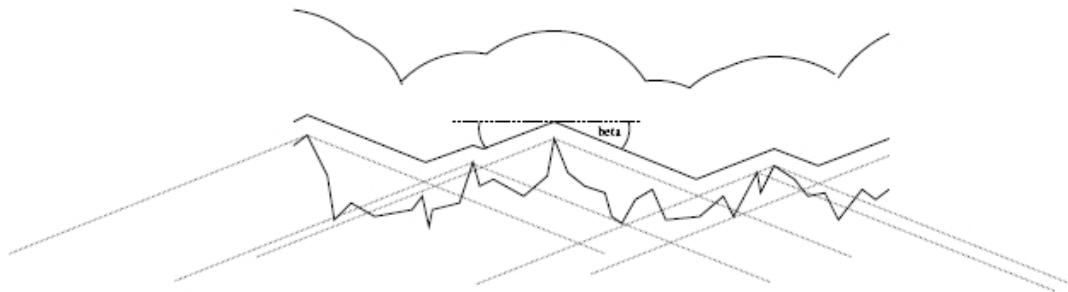
dove

$$f_x(q) = s(q) + \sqrt{r^2 - (q - x)^2} \quad , \quad q \in [x - r, x + r] \quad . \quad (3.2)$$

La curva ideale viene poi calcolata da questi punti di contatto. Una tecnica



**Figura 3.2:** Tracciato della curva di offset effettiva risultante dalla superficie  $s(x)$ .

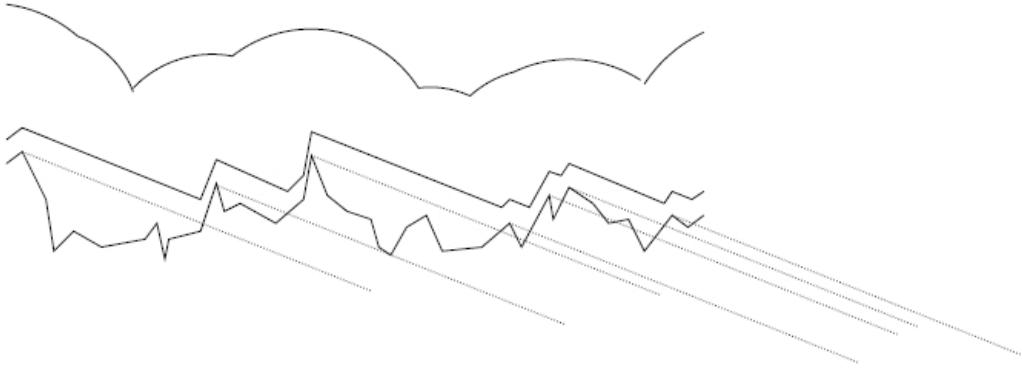


**Figura 3.3:** Approssimazione del tracciato compiuto dalla palla durante il rotolamento.

più semplice (e quindi anche meno dispendiosa in termini di risorse di calcolo) è rappresentata in figura 3.3. La traiettoria in figura 3.2 converge alla curva ideale di figura 3.3 per raggi molto grandi se comparati alla ruvidità della superficie. Infatti, in una prima implementazione, anche le forti semplificazioni (riportate figura 3.4) realizzate con un algoritmo molto semplice, hanno dato risultati convincenti.

### 3.1.2 La superficie

Esistono diverse tecniche per realizzare il profilo della superficie alla base del modello di rotolamento. Una possibilità è quella di campionare o effettuare una scansione di superfici reali e usare tali segnali come input per lo stadio seguente del modello; questo approccio però non si adatta ai nostri obiettivi: noi siamo interessati ad un modello parametrico, flessibile ed efficiente piuttosto che ad



**Figura 3.4:** Ulteriore approssimazione del tracciato di rotolamento.

una singola simulazione realistica. Inoltre i segnali memorizzati sono difficili da adattare alle variazioni degli attributi del modello; preferiamo quindi usare modelli statistici di superfici che possano efficientemente generare segnali per i vari attributi.

E' comune nella computer graphics descrivere le superfici tramite metodi frattali. L'applicazione di questa idea al nostro modello unidimensionale conduce all'utilizzo di un segnale di rumore con spettro di potenza  $1/f^\beta$ , o equivalentemente rumore bianco filtrato con queste caratteristiche. Il parametro reale  $\beta$  riflette la dimensione frattale (o ruvidità). I risultati pratici di questo tipo di modello sono diventati più convincenti quando la banda del segnale della superficie è stata fortemente limitata; ciò non deve sorprendere se pensiamo che solitamente le superfici coinvolte nel rotolamento sono molto smussate. Smussare su larga scala (che può essere assimilato al tagliare pezzi di pietra per pavimentazioni) corrisponde ad un filtraggio passa-alto, mentre smussare a livello microscopico (come lucidare una pietra) può essere visto come un filtraggio di tipo passa-basso. Tramite queste elaborazioni però si possono perdere le caratteristiche del rumore  $1/f^\beta$  di partenza. Perciò optiamo per una approssimazione di questa curva con un filtro del secondo ordine la cui ripidità è proporzionale al grado di ruvidità a livello microscopico.

Tutte le frequenze in questo modello di basso livello devono variare proporzionalmente ai parametri di velocità, perciò l'ampiezza del segnale di superficie deve essere mantenuta costante. Naturalmente i parametri dell'impatto, in particolare la costante di elasticità  $k$ , devono essere variati opportunamente a seconda della superficie che si vuole simulare (cioè in base alle proprietà del materiale), in quanto contribuiscono fortemente alla espressività del modello.

### 3.1.3 Il modello di impatto

Un suono di contatto è descritto tramite due sistemi, uno per l'oggetto risonante e uno per l'oggetto percussore. Supposto che la superficie di contatto sia piccola, la forza di contatto viene espressa come:

$$f(x(t), v(t)) = \begin{cases} kx(t)^\alpha + \lambda x(t)^\alpha \cdot v(t) = kx(t)^\alpha(1 + \mu v(t)) & x > 0 \\ 0 & x \leq 0 \end{cases}, \quad (3.3)$$

dove  $v(t) = \dot{x}(t)$  è la velocità di compressione,  $k$  è il coefficiente di rigidità,  $\alpha$  è un parametro che descrive la geometria locale dell'impatto (nel caso di due perfette sfere vale 1.5),  $\lambda$  è un coefficiente di smorzamento e  $\mu = \lambda/k$  è un termine matematico (senza significato fisico) detto *caratteristica viscoelastica*.

Il percussore è considerato una massa ideale, quindi l'unico parametro che lo caratterizza è la massa; il risonatore invece è un oggetto modale ed è caratterizzato dai parametri di frequenza, tempi di decadimento,  $k$ ,  $\alpha$  e  $\lambda$ . Si assume inoltre che il percussore abbia un elevato coefficiente di smorzamento: in tal modo diventa trascurabile l'energia acustica delle sue vibrazioni, e l'energia viene trasferita al risonatore che emette il suono. Per una descrizione matematica vengono sintetizzati i modi di vibrazione (teoricamente infiniti), ognuno dei quali fornisce un contributo allo spettro del segnale [1].

### 3.1.4 Caratteristiche di alto livello

Oltre ai parametri di basso livello visti nella sezione precedente, i tipici moti di rotolamento posseggono caratteristiche a livello macroscopico che contribuiscono fortemente alla percezione acustica, e non possono essere descritti come fatto in precedenza. Molte superfici contengono dei pattern più o meno regolari che non possono essere classificati come rumore frattale filtrato, e tali periodicità possono essere verificate attraverso l'esperienza di tutti i giorni: i pavimenti in pietra, o i solchi pseudoperiodici in molte tavole di legno. Le singole irregolarità sulla superficie dell'oggetto rotolante possono essere raggruppate in una sola categoria, dal momento che sono richiamate periodicamente nel movimento rotatorio. Tale caratteristica può essere modellata con segnali impulsivi di frequenza costante o variante in un piccolo intervallo; potrebbero essere utili delle approssimazioni sinusoidali o polinomiali, con un parametro di smussamento legato al grado di approssimazione della funzione. Ancora, le frequenze devono variare proporzionalmente alla velocità.

Dev'essere fatta un'altra osservazione a livello macroscopico: per oggetti rotolanti che non sono perfettamente sferici (in maniera rilevante per il movimento)

la velocità del punto di contatto su entrambe le superfici e l'effettiva forza che preme l'oggetto rotolante sulla superficie variano periodicamente; devono essere variati questi due parametri per modellare tale deviazione dalla sfericità perfetta.

Infine notiamo che, come nell'ascolto di tutti i giorni, gli scenari acustici del rotolamento di oggetti sono riconosciuti e accettati più facilmente se sono presenti dinamiche tipiche; ad esempio pensiamo al suono di una palla che cade e che rimbalza fino a quando non raggiunge un contatto costante con il suolo: a questo punto il rotolamento diventa chiaro dal punto di vista uditivo e la velocità media lentamente diminuisce fino diventare nulla.

## 3.2 Tessiture della superficie

Molti dei suoni di contatto ai quali siamo interessati non possono essere ricreati in modo convincente usando solo modelli deterministici, come nel caso dei suoni di rotolamento risultanti dalla sequenza di micro impatti tra due oggetti risonanti, determinati dal profilo della superficie di contatto. Affrontiamo quindi il problema di effettuare il rendering delle tessiture di superfici attraverso processi frattali; tali processi sono molto usati nella computer graphics, dal momento che forniscono tessiture che sembrano naturali all'occhio umano. Dato che nei modelli fisici le proprietà delle superfici vengono tradotte direttamente in segnali di forza e, di conseguenza, in suoni, sembra naturale seguire lo stesso approccio per modellare le superfici.

I frattali sono definiti [17] come geometrie invarianti rispetto alla scalatura. Sono auto-simili se la scalatura è isotropica o uniforme in tutte le direzioni, auto-affini se la scalatura è anisotropica o dipendente dalla direzione, staticamente auto-simili se sono l'unione di copie di se stessi scalate statisticamente. Più formalmente, un processo frattale unidimensionale può essere definito come una generalizzazione della definizione di moto standard Browniano [38]. Il processo stocastico  $x = \{x(t), t \geq 0\}$  è un moto standard Browniano se:

1. il processo stocastico  $x$  ha incrementi indipendenti;
2. vale la proprietà

$$x(t) - x(s) \sim N(0, t - s) \quad \text{per} \quad 0 \leq s < t;$$

cioè l'incremento  $x(t) - x(s)$  è normalmente distribuito con media nulla e varianza  $(t - s)$ ;

3.  $x(0) = 0$ .

La definizione di moto standard Browniano può essere generalizzata alla definizione di *processo frattale* se l'incremento  $x(t) - x(s)$  è normalmente distribuito con media 0 e varianza proporzionale a  $(t - s)^{2H}$ . Il parametro  $H$  è chiamato *esponente di Hurst* e caratterizza il comportamento del processo frattale rispetto alla scalatura: se  $x = \{x(t), t \geq 0\}$  è un processo frattale con esponente di Hurst  $H$ , allora, per ogni reale  $a > 0$ , obbedisce alla seguente relazione di scala:

$$x(t) \stackrel{P}{=} a^{-H} x(at) , \quad (3.4)$$

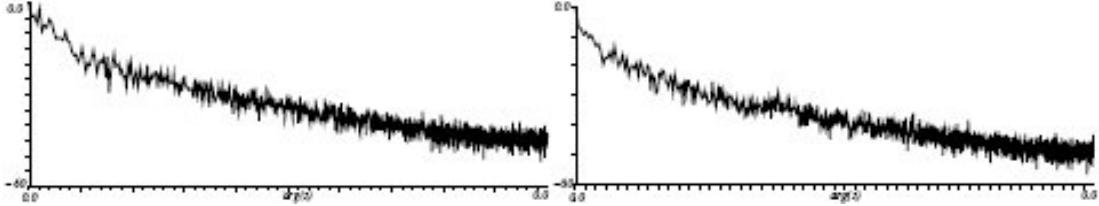
dove  $\stackrel{P}{=}$  denota l'uguaglianza statistica. Questa è la definizione formale di *auto-similirarità statistica*. La famiglia di processi 1/f statisticamente auto-simili, nota anche come rumore 1/f, è composta da processi aventi densità di spettro di potenza  $S_x(\omega)$  proporzionale a  $1/\omega^\beta$ , con  $\beta$  legato all'esponente di Hurst  $H$  dalla relazione  $\beta = 2H + 1$ . Per  $\beta = 0$  la definizione corrisponde al rumore bianco, per  $\beta = 2$  si ottiene il rumore Browniano, e per  $\beta = 1$  il rumore risultante è rumore rosa. Il parametro  $\beta$  è in relazione anche con la dimensione frattale. La dimensione frattale [47] di una funzione è un parametro reale che determina l'irregolarità di un oggetto frattale, è legata al grafico della funzione ed è usata nella computer graphics per controllare la ruvidità percepita [35]. Per i processi 1/f, tale dimensione è inversamente proporzionale all'esponente di Hurst  $H$ : valori maggiori di  $H$  corrispondono a valori minori della dimensione frattale;  $H$  è proporzionale a  $\beta$ . Perciò, incrementando  $\beta$  possiamo raggiungere una redistribuzione della potenza dalle alte alle basse frequenze, con uno smussamento complessivo della forma d'onda.

Il problema di generare il rumore 1/f è stato trattato estensivamente. Uno degli approcci più comuni risulta quello di filtrare una sorgente di rumore bianco per ottenere lo spettro 1/f; seguendo questo procedimento utilizzeremo il modello riportato in [39] e [5]. Il filtro è una cascata di  $N$  filtri del primo ordine, ognuno con una coppia di poli e zeri; la funzione di trasferimento  $H(s)$  nel dominio di Laplace è la seguente:

$$H(s) = A \frac{\prod_{i=1}^N (s - s_{0i})}{\prod_{i=1}^N (s - s_{pi})} , \quad (3.5)$$

dove  $A$  è una costante. Il generatore di rumore frattale è ottenuto impostando opportunamente i poli e gli zeri dei filtri nella cascata [39]. In particolare, il polo e lo zero alle frequenze  $f_{pi}$  e  $f_{0i}$  possono essere computati come funzioni di  $\beta$  con le seguenti formule:

$$f_{pi} = -\frac{s_{pi}}{2\pi} = f_{p(i-1)} 10^{\frac{1}{h}} , \quad (3.6)$$



**Figura 3.5:** Spettro di ampiezza del rumore frattale generato con  $\beta = 1.81$ ,  $h = 2$  a sinistra e  $h = 6$  a destra.

$$f_{0i} = -\frac{s_{0i}}{2\pi} = f_{p1} 10^{\frac{\beta}{2h}} \quad , \quad (3.7)$$

dove  $f_{p1}$  è il polo di frequenza più bassa del filtro; perciò il limite inferiore della banda di frequenza per l'approssimazione è  $f_{p1}$ . La densità  $h$  (densità dei poli per decade di frequenze) può essere usata per controllare l'errore tra lo spettro desiderato e lo spettro approssimato ottenuto dal filtraggio del rumore bianco. La dipendenza dell'errore in relazione alla densità dei poli del filtro è discussa in [5]. La figura 3.5 mostra uno spettro  $1/f^\beta$  ottenuto usando il filtro 3.5, con due diversi valori per  $h$ .

La funzione di trasferimento nel dominio discreto del tempo può essere computata con il metodo della varianza della risposta all'impulso [33]; ciò corrisponde a mappare poli e zeri della funzione di trasferimento  $H(s)$  su poli e zeri della funzione di trasferimento  $H(z)$  nel dominio discreto del tempo attraverso la seguente sostituzione:

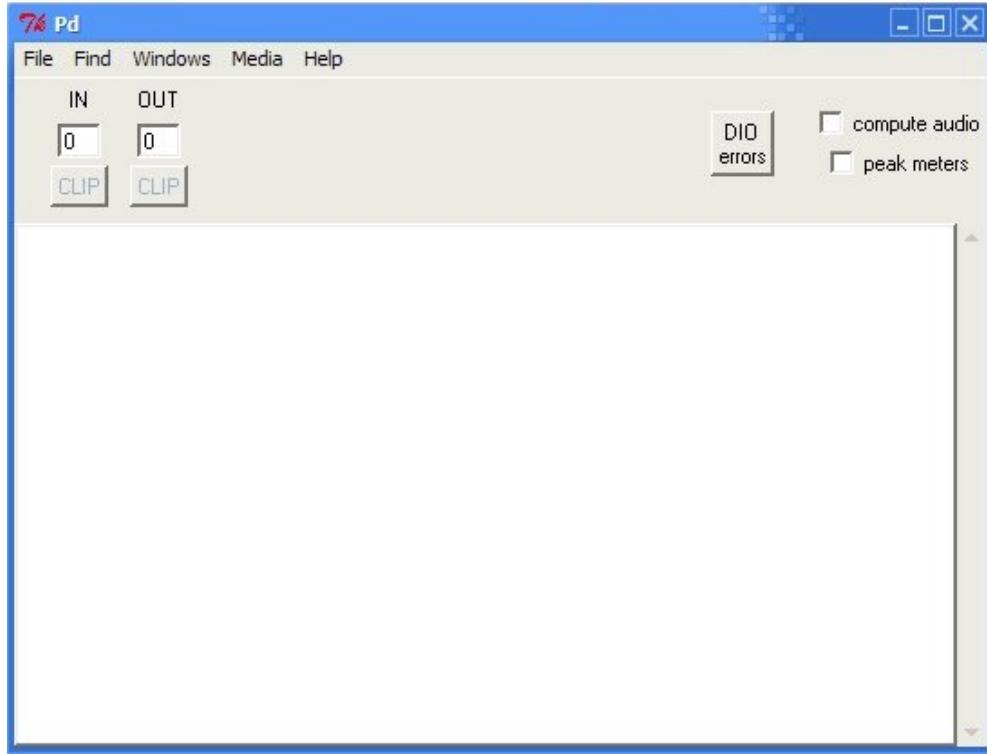
$$s - s_x \rightarrow 1 - e^{s_x T_s} z^{-1} \quad , \quad (3.8)$$

dove  $T_s$  è il periodo di campionamento e  $s_x$  indica un polo  $s_{pi}$  o uno zero  $s_{0i}$ . Si ottiene la seguente funzione di trasferimento discreta:

$$H(z) = A' \frac{\prod_{i=1}^N 1 - e^{s_{0i} T_s} z^{-1}}{\prod_{i=1}^N 1 - e^{s_{pi} T_s} z^{-1}} \quad , \quad (3.9)$$

dove  $A'$  è una costante di normalizzazione. In conclusione, lo spettro  $1/f^\beta$  è approssimato da una cascata di filtri del primo ordine, ognuno con la seguente funzione di trasferimento discreta:

$$H^{(i)}(z) = \frac{1 + b_i z^{-1}}{1 + a_i z^{-1}} \quad , \quad \text{con} \quad \begin{cases} a_i = e^{-2\pi f_{pi} T_s}, & b_i = e^{-2\pi f_{0i} T_s} \\ f_{pi} = f_{p(i-1)} 10^{\frac{1}{h}}, & f_{0i} = f_{p(i-1)} 10^{\frac{\beta}{2h}} \end{cases} \quad (3.10)$$



**Figura 3.6:** L'interfaccia grafica di Pure Data.

### 3.3 Implementazioni dei modelli in Pure Data

#### 3.3.1 Cos'è Pure Data

Pure Data è un software ideato da Miller Puckette: si tratta di un ambiente di programmazione visuale in real-time per l'elaborazione di audio e grafica, basato sul sistema *Max/MSP* ma più semplice e portatile di questo. Sono presenti due caratteristiche in PD molto importanti: la possibilità di gestire contemporaneamente la simulazione video e la simulazione audio utilizzando il pacchetto *GEM* di Mark Dank e delle facilitazioni nelle definizioni nell'accesso alle strutture dati.

Ogni documento di PD è chiamato *patch*; una volta che tale file viene aperto si presenta composto di una finestra principale e di eventuali sotto-finestre (che possono essere visualizzate o nascoste ma sono sempre in esecuzione). In ogni finestra compaiono dei blocchi collegati tra loro; i blocchi possono essere di quattro tipi:

**Oggetti** – Un oggetto viene creato scrivendo del testo all'interno del blocco; il testo viene diviso in *atomi*: il primo atomo definisce il tipo di oggetto che viene creato, i successivi costituiscono gli argomenti di creazione, i quali servono ad inizializzare l'oggetto.

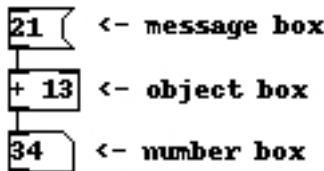
Ogni oggetto può possedere zero o più *inlet* (collegamenti in input) e zero o più *outlet* (collegamenti in output); il numero di questi dipende dal tipo

di oggetto. Ci sono due tipi di collegamento: *collegamenti di segnale* e *collegamenti di controllo*; i primi sono rappresentati da una linea marcata, mentre i secondi da una linea sottile. La scelta del tipo di collegamento dipende dall'outlet dal quale provengono; un outlet può essere collegato ad un inlet solo se entrambi accettano collegamenti dello stesso tipo (o entrambi di segnale o entrambi di controllo).

In figura è riportato un esempio di oggetto: l'atomo + definisce la tipologia di oggetto (un blocco sommatore), mentre il secondo atomo 13 indica il valore da sommare all'ingresso.



**Messaggi** – I blocchi di messaggio interpretano il testo come un messaggio da inviare ogni volta che il blocco viene attivato; l'invio è verso il blocco al quale l'outlet è collegato e può avvenire un numero qualsiasi di volte durante l'esecuzione della patch. Il blocco di messaggio possiede sempre un inlet e un outlet. Nell'esempio seguente il primo blocco, quando viene attivato dal click del mouse, invia il messaggio 21 all'oggetto che lo sommerà a 13; all'ultimo blocco viene inviato il risultato dell'operazione. Un messaggio



**Figura 3.7:** Semplice esempio di patch per Pure Data.

può essere attivato cliccandoci sopra, da un altro messaggio in ingresso o da un particolare blocco chiamato *bang*.

**GUI** – Il terzo blocco dell'esempio precedente fa parte dei blocchi GUI (graphical user interface); tra questi sono inclusi i blocchi numerici, blocchi contenenti simboli, controlli scorrevoli e pulsanti. Mentre gli oggetti rimangono immutati durante l'esecuzione di una patch, i blocchi GUI aggiornano il loro stato in base al valore che contengono.

**Commenti** – I commenti sono costituiti da semplice testo e non sono contenuti all'interno di nessun rettangolo. In figura 3.7 i testi alla destra dei blocchi sono commenti.

Una patch può essere in modalità *edit* oppure in modalità *running*: nel primo caso la patch non è in esecuzione ed è permessa la creazione o modifica dei blocchi

e dei collegamenti; nel secondo caso la patch è in esecuzione, è possibile ancora modificare i collegamenti, mentre la modifica dei blocchi GUI ha l'effetto di variare i parametri di controllo della patch.

### L'incapsulamento in Pure Data

Come avviene con i linguaggi di programmazione quali C, C++ e Java, con Pure Data è possibile scrivere del codice che può poi essere riutilizzato in qualsiasi momento; uno o più oggetti infatti possono essere costituiti da *subpatch*, ovvero delle patch separate che vengono incapsulate all'interno dell'oggetto. Si possono distinguere due tipi di incapsulamento:

**one-off subpatch** – se l'oggetto viene chiamato `pd` o `pd my-name`, viene creata una subpatch il cui contenuto viene salvato come parte della patch genitore che può essere riutilizzata e modificata più volte all'interno di quest'ultima;

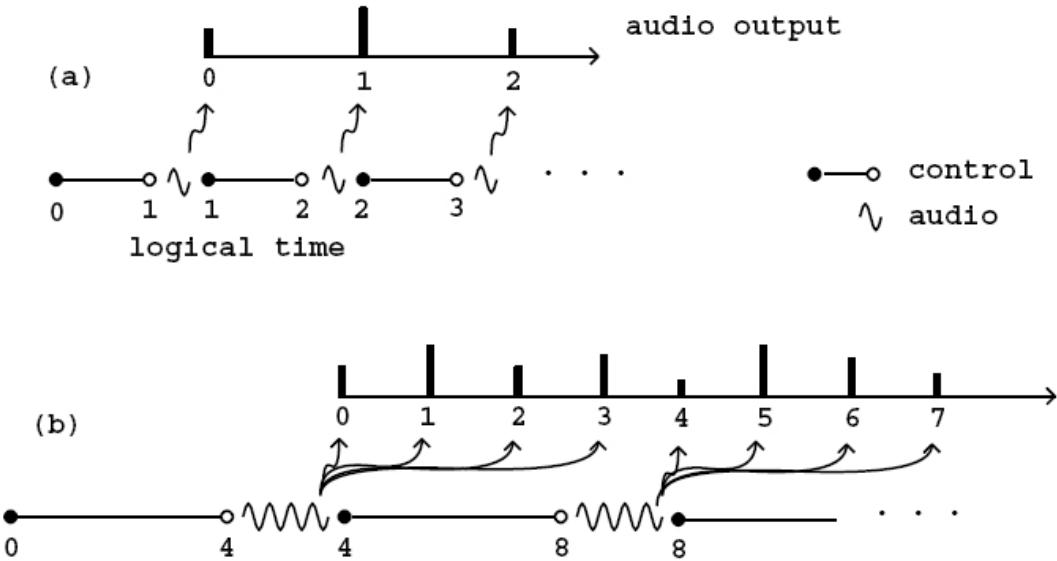
**astrazione** – se l'oggetto ha il nome di una patch già presente come file (omettendo l'estensione `.pd`), PD caricherà il contenuto del file all'interno della subpatch; in tal caso un cambiamento alla patch si propaga a tutte le chiamate alla sua astrazione.

Per definire il numero di inlet e outlet che deve possedere l'oggetto contenente la subpatch è sufficiente utilizzare all'interno di quest'ultima i blocchi `inlet` e `outlet` (oppure `inlet~` e `outlet~` per i collegamenti di segnale).

### 3.3.2 Gestione dei segnali audio

In Pure Data i segnali audio vengono memorizzati come numeri in virgola mobile a 32 bit; a seconda dell'hardware utilizzato però l'output viene limitato a 16 o 24 bit. L'input è sempre compreso tra i valori 1 e -1, mentre l'output viene tagliato al fine di restare compreso tra questi due limiti. La frequenza di campionamento di default è 44100 Hz (modificabile da riga di comando o nel menù *audio setup*).

Le computazioni audio vengono eseguite dai *blocchi tilde*, cioè quelli che, per convenzione, hanno il nome seguito da una tilde, come `osc~`; essi comunicano attraverso connessioni di segnale. All'avvio della computazione, o quando vengono cambiati i collegamenti, gli oggetti tilde vengono ordinati secondo un ordine di esecuzione lineare; tale lista viene poi eseguita in blocchi di 64 campioni ciascuno (a 44.1 KHz significa che l'intera rete di blocchi audio viene eseguita una volta ogni 1.45 millisecondi). Le connessioni nella rete audio devono essere acicliche; la presenza di eventuali cicli viene rilevata al momento del riordino dei blocchi.



**Figura 3.8:** Linee del tempo per la computazione audio e la computazione di controllo in Pure Data con (a) blocchi di un campione e (b) blocchi di quattro campioni.

Ogni subpatch può avere dei collegamenti di segnale in entrata e in uscita tramite i blocchi “inlet~” e “outlet~”.

La computazione dei segnali non avviene in *real time*, ma in *logical time*: quest’ultimo è definito come l’istante del successivo campione audio che verrà elaborato, ed è sempre precedente al real time, definito come l’istante in cui il campione arriva all’output. Tutto questo serve a far sì che la computazione audio sia indipendente dal tempo effettivo di esecuzione del processore, il quale può variare per molte ragioni. Si può dedurre che una computazione audio, se eseguita nel modo corretto, è deterministica: due esecuzioni dello stesso calcolo, una in tempo reale e l’altra no, devono dare lo stesso risultato. In figura 3.8 si vede come la computazione dell’audio viene svolta rispetto all’elaborazione dei segnali di controllo: i campioni audio vengono calcolati a scadenze regolari, ma prima di ogni scadenza devono essere effettuati tutti i calcoli di controllo che possono influenzare il campione audio in quella scadenza. Se  $N$  è il numero di campioni in un blocco, la prima computazione audio riguarda i campioni da 0 a  $N - 1$ , i quali vengono inviati in output tutti insieme all’istante  $N$  (logical time); prima di questo istante vengono effettuate tutte le elaborazioni di controllo per gli  $N$  campioni.

### Conversione tra segnali audio e segnali di controllo

La conversione da segnale di controllo a segnale audio è possibile utilizzando l’oggetto `sig~`. Per quanto riguarda la conversione inversa (da segnale a controllo)

deve essere specificato l’istante nel quale il segnale viene campionato; questo può essere gestito tramite l’oggetto **snapshot~** che campiona il segnale ogni volta che riceve in input un *bang*. Gli oggetti **+~**, **-~**, **\*~**, **/~**, **osc~** e **phasor~** possono essere configurati per accettare entrambi i tipi di segnale.

### Selettori e blocchi

Gli oggetti **switch~** e **block~** sono utilizzati per attivare o disattivare parti della computazione audio e per controllare la dimensione dei blocchi di calcolo; deve essere presente uno solo dei due oggetti per ogni finestra della patch e il suo effetto verrà esteso a tutte le subpatch. Entrambi accettano due argomenti per la loro costruzione: il primo è la dimensione del blocco e il secondo un fattore di sovrapposizione.

L’oggetto **switch~** può essere usato per ridurre il carico computazionale scegliendo, ad esempio, uno tra diversi algoritmi di sintesi da utilizzare: per farlo è sufficiente che ogni algoritmo sia implementato in una subpatch diversa.

### Connessioni esterne

I segnali possono essere inviati non solo tra blocchi di una stessa finestra, ma anche tra finestre diverse oppure possono essere dati in input all’algoritmo che li ha generati in una configurazione in retroazione. Questo può essere implementato attraverso tre coppie di oggetti:

**throw~/catch~** – **throw~** accumula dati in un bus, mentre **catch~** legge i dati accumulati e riazzera il bus per il ciclo successivo;

**send~/receive~** – **send~** salva un segnale che può essere ricevuto più volte da un blocco **receive~**, il quale però può leggere un solo **send~** alla volta;

**delread~/delwrite~** – se viene inviato un segnale ad un punto precedente nella rete audio, esso viene ricevuto solo al ciclo successivo, con un ritardo quindi di 1.45 millisecondi (con le impostazioni di default). Gli oggetti **delread~** e **delwrite~** permettono di ridurre al minimo tale ritardo.

### Scheduling

Lo scheduler di Pure Data cerca di mantenere un certo *vantaggio* sul calcolo in modo da poter assorbire eventuali forti incrementi nel carico computazionale; tale comportamento può essere impostato tramite le flag “audiobuffer” o “frags”.

Se durante l’elaborazione dell’audio si accumulano dei ritardi, possono verificarsi delle interruzioni nei flussi di input e output; tuttavia lo streaming su disco non viene influenzato.

Le operazioni di PD sono deterministiche, nel senso che le computazioni vengono eseguite nel momento in cui vengono schedulate senza subire cambiamenti di ordine in real-time. Se un'operazione viene attivata da un evento esterno, viene associata ad un tempo; questo serve a garantire che le esecuzioni siano consistenti con le scadenze temporali imposte dallo scheduler (il tempo non deve mai decrescere).

### 3.3.3 Scrivere *external* per Pure Data

Con il termine *external* si indica un oggetto che non è compreso in Pure Data ma che può essere caricato dinamicamente durante l'esecuzione di PD; si differenziano dagli *internal* in quanto questi ultimi sono le primitive già incluse in PD. Una volta che un external viene caricato in memoria, non è più distinguibile dagli internal. Una libreria è una collezione di external compilati all'interno di un unico file binario; il nome di una libreria varia a seconda del sistema operativo per la quale viene implementata: ad esempio, se viene creata la libreria `my_lib`, essa dovrà essere chiamata `my_lib.pd_linux` nei sistemi Linux, `my_lib.pd_irix` e `my_lib.dll` nei sistemi Win32. Una libreria elementare include esattamente un external avente lo stesso nome della libreria.

A differenza degli external, una libreria può essere importata in due modi:

- tramite opzione da riga di comando: `-lib my_lib` (così la libreria e tutti gli external in essa contenuti vengono caricati all'avvio di PD);
- creando un oggetto `my_lib` (consigliabile quando la libreria contiene un solo oggetto con il nome della libreria stessa).

In entrambi i casi PD prima controlla se una libreria `my_lib` è già stata caricata; se così non è, viene cercato il file corrispondente e, se trovato, tutti gli external inclusi vengono caricati.

Pure Data è scritto in C, quindi anche gli external vanno scritti in questo linguaggio di programmazione; il codice per un semplice external che stampa il messaggio “hello world!” è riportato di seguito [50]:

```
#include "m_pd.h"

static t_class *helloworld_class;

typedef struct _helloworld {
    t_object x_obj;
} t_helloworld;
```

```
void helloworld_bang(t_helloworld *x)
{
    post("Hello world !!");
}

void *helloworld_new(void)
{
    t_helloworld *x = (t_helloworld *)pd_new(helloworld_class);
    return (void *)x;
}

void helloworld_setup(void)
{
    helloworld_class = class_new(gensym("helloworld"),
                                (t_newmethod)helloworld_new,
                                0, sizeof(t_helloworld),
                                CLASS_DEFAULT, 0);
    class_addbang(helloworld_class, helloworld_bang);
}
```

Inizialmente viene definita la nuova *classe* (qui il termine “classe” viene usato con un significato diverso da quello usuale della programmazione ad oggetti), dove `hello_worldclass` è un puntatore alla nuova classe e la struttura `t_helloworld` costituisce il *dataspace* della classe; la variabile `t_object` è assolutamente necessaria e serve a memorizzare le proprietà dell’oggetto, come la sua rappresentazione grafica e le informazioni su inlet e outlet.

Vengono poi definite le funzioni (*methods*) per manipolare i dati; quando l’istanza della classe riceve un dato, viene richiamato un metodo; ogni metodo è associato ad un inlet. La funzione implementata viene eseguita solo quando un nuovo dato arriva a tale inlet.

Al caricamento della libreria, PD richiama la funzione `helloworld_setup()`: tale funzione dichiara la nuova classe e le sue proprietà. L’istruzione `class_new` crea una nuova classe e ritorna un puntatore ad essa: il primo argomento è il nome simbolico della classe; il secondo e il terzo definiscono il costruttore e il distruttore; il quarto definisce la dimensione della struttura dati; il quinto determina l’aspetto grafico dell’oggetto; i rimanenti sono gli argomenti dell’oggetto. L’istruzione successiva serve per aggiungere i metodi alla classe (il primo argomento è la classe, il secondo è il metodo).

L’inizializzazione dell’oggetto avviene tramite la funzione `helloworld_new()`, i cui argomenti dipendono dalla definizione data con `class_new`.

### 3.3.4 Librerie utili per Pure Data

#### GEM

GEM (acronimo per *Graphical Environment for Multimedia*, <http://gem.iem.at>) è una collezione di external che permettono di integrare elaborazioni grafiche OpenGL in una patch; sono disponibili diversi tipi di forme geometriche, di luci e di texture; è possibile implementare il movimento della visuale e processare l'immagine.

Le elaborazioni della parte audio e della grafica vengono svolte contemporaneamente: in tal modo si può creare un vero e proprio scenario virtuale semplicemente utilizzando una rete di blocchi creati e gestiti come i blocchi nativi di PD.

#### Flext

Si è visto che Pure Data è un software scritto in C, e gli external devono essere scritti in tale linguaggio; l'utente però potrebbe avere la necessità di usare le meno complesse strutture del C++, nonché il pieno supporto alla programmazione ad oggetti che questo offre. Per soddisfare questa esigenza nasce *flext* (<http://grrrr.org/ext/flext/>), una libreria per lo sviluppo di external in C++. Con flext è possibile creare librerie di external che possono essere compilate per Pure Data, per Max/MSP e per differenti piattaforme (Windows, Linux, OSX) e compilatori.

Un semplice esempio di external basato su flext è il seguente.

```
// inclusione del header file
#include <flext.h>

// controllo sulla versione
#if !defined(FLEXT_VERSION) || (FLEXT_VERSION < 400)
#error You need at least flext version 0.4.0
#endif

// definizione della classe
// Attenzione: il nome della classe deve essere lo stesso
// nome dell'oggetto (senza l'eventuale ~)
class simple1:

public flext_base
{
```

```
FLEXT_HEADER(simple1,flext_base)

public:
// costruttore
simple1()
{
    // definizione degli inlets:
    // il primo deve essere sempre di tipo anything
    // (oppure signal per gli oggetti dsp)
    AddInAnything();

    // definizione degli outlets:
    AddOutFloat(); // aggiunta di un outlet float (indice 0)

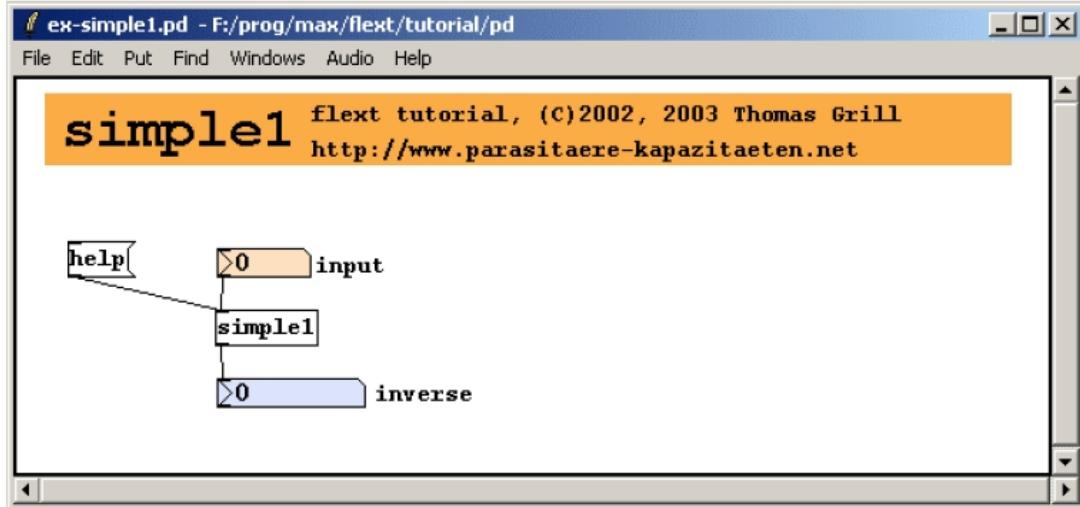
    // registrazione dei metodi:
    // registra il metodo "m_float" per l'inlet 0
    FLEXT_ADDMETHOD(0,m_float);
}

protected:
// definizione del metodo
void m_float(float input)
{
    float result;
    if(input == 0) {
        post("%s - zero can't be inverted!",thisName());
        result = 0;
    }
    else
        result = 1/input;

    // manda il valore in output all'outlet
    ToOutFloat(0,result); // (0 è l'indice dell'outlet)
}

private:
// callback per il metodo "m_float"
FLEXT_CALLBACK_1(m_float,float)
};

// creazione della classe
```



**Figura 3.9:** Utilizzo del modulo simple1 basato su *flext*: accetta un numero in input e invia in output il suo inverso.

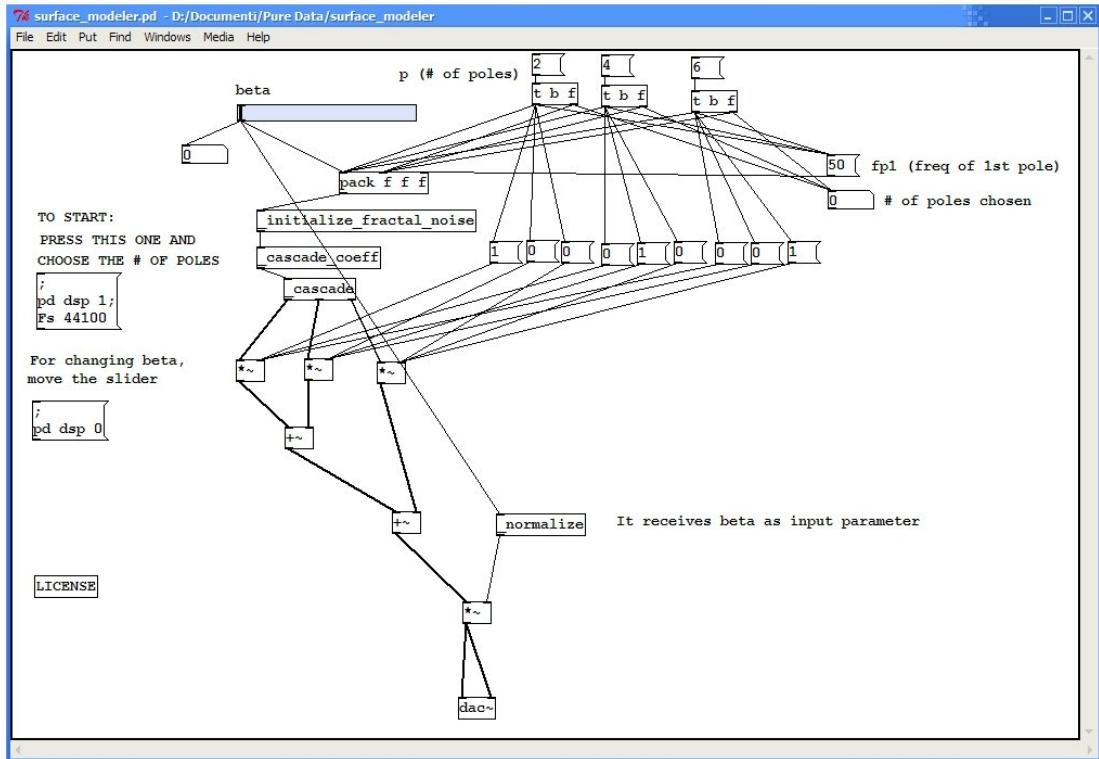
```
FLEXT_NEW("simple1",simple1)
```

Come si può notare, è stata utilizzata la programmazione ad oggetti, a partire dalla creazione di una classe derivata dalla classe base *flext\_base*, la quale contiene tutte le funzioni essenziali. Il costruttore viene richiamato nel momento in cui l'oggetto è incluso nella patch; più precisamente è chiamato quando si crea un'istanza della classe, contiene tutte le inizializzazioni necessarie e ha lo stesso nome della classe. Tra le inizializzazioni devono essere presenti le dichiarazioni di inlet e outlet e le associazioni tra metodi e rispettivi inlet.

Un *callback wrapper* è necessario per stabilire un collegamento con PD per ogni metodo che deve essere lanciato ogni volta che un dato viene ricevuto: ciò avviene tramite l'istruzione `FLEXT_CALLBACK_1(m_float, float)`. Con l'ultimo comando si informa il sistema riguardo al nome della classe e ai suoi argomenti di creazione. In figura 3.9 è riportato un esempio di utilizzo di questo external.

### 3.3.5 Implementazione della patch generatrice di rumore frattale

Nella realizzazione della patch per Pure Data che implementa l'algoritmo di generazione di rumore frattale, per convenienze implementative, i filtri sono stati riscritti come cascata di biquadri: perciò la cascata è formata da  $N/2$  filtri del secondo ordine, ognuno con le seguenti funzioni di trasferimento (calcolate



**Figura 3.10:** La patch *surface\_modeler* che implementa la generazione di rumore frattale.

dall'equazione 3.10):

$$\begin{aligned}
 H^{(i)}(z) = H^j H^{j-1}(z) &= \frac{(1 + b_j z^{-1})(1 + b_{j-1} z^{-1})}{(1 + a_j z^{-1})(1 + a_{j-1} z^{-1})} \\
 &= \frac{1 + (b_j + b_{j-1})z^{-1} + (b_j b_{j-1})z^{-2}}{1 + (a_j + a_{j-1})z^{-1} + (a_j a_{j-1})z^{-2}}
 \end{aligned}$$

con  $j = 2 \cdot i, i = 1 \dots N/2$ .

Il parametro di controllo più importante impostabile dall'utente è  $\beta$ , che definisce lo spettro  $f^\beta$ ; deve essere impostato anche il numero di poli della cascata di filtri assieme alla frequenza del primo polo: con questi parametri viene controllata l'accuratezza dell'approssimazione  $1/f^\beta$ .

In figura 3.10 è riportata la patch *surface\_modeler* che implementa la generazione di rumore frattale. Per avviare la computazione è sufficiente cliccare sul blocco (in modalità *running*):

```

;
pd dsp 1;
Fs 44100

```

subito dopo si seleziona il numero di poli desiderato (due, quattro o sei). Il

controllo sulla patch avviene variando il parametro  $\beta$  tramite lo slider:



Il modulo `_initialize_fractal_noise` rappresenta una subpatch (figura 3.11) tramite la quale vengono effettuati i calcoli in 3.6 e 3.7.

Il modulo `_cascade` invece è una subpatch nella quale viene implementata una cascata di tre filtri, come è possibile vedere in figura 3.12; ogni oggetto `biquad~` è un filtro biquadro a due poli e due zeri; ognuno di questi filtri calcola le seguenti equazioni differenziali:

$$\begin{aligned} y(n) &= ff1 \cdot w(n) + ff2 \cdot w(n-1) + ff3 \cdot w(n-2) \\ w(n) &= x(n) + fb1 \cdot w(n-1) + fb2 \cdot w(n-2) \end{aligned} \quad (3.11)$$

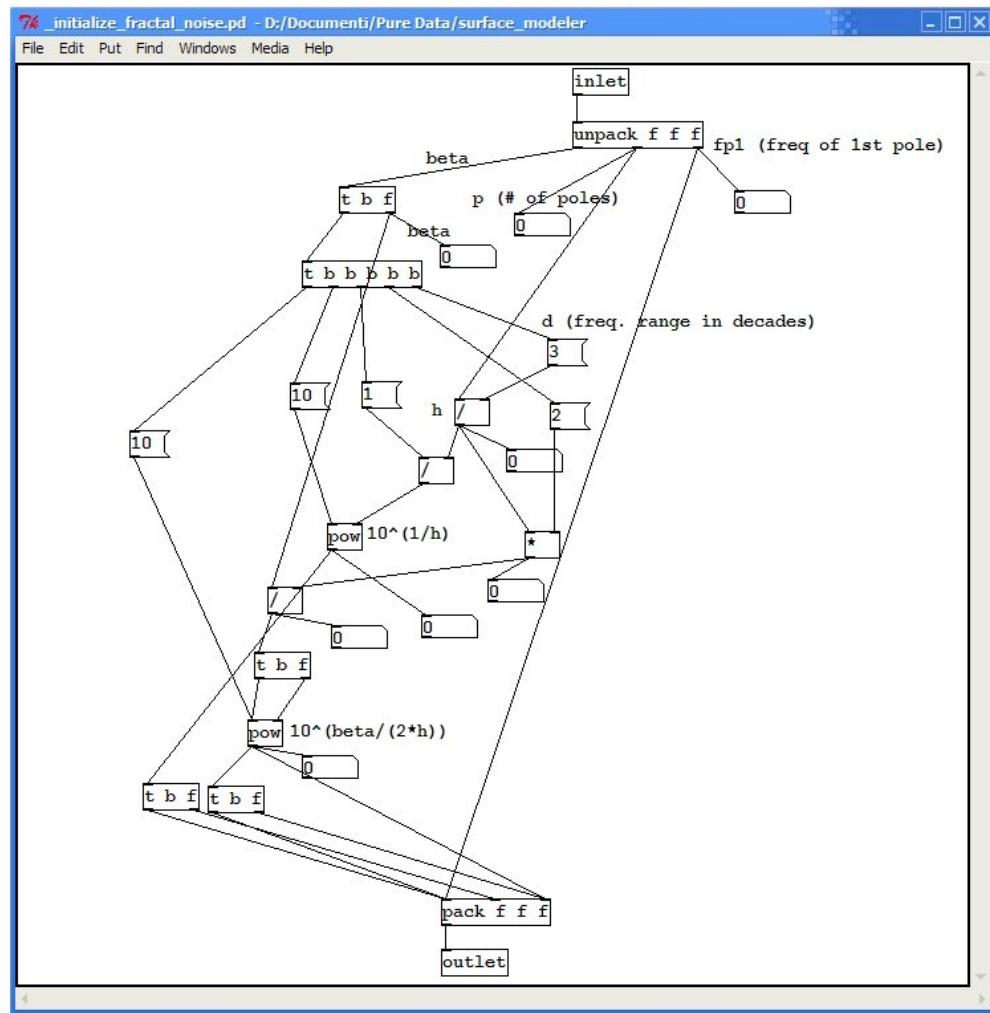
I valori  $fb1, fb2, ff1, ff2, ff3$  vengono dati, in quest'ordine, come argomenti di creazione dell'oggetto.

### 3.3.6 Implementazione della patch generatrice di rumore di sfregamento

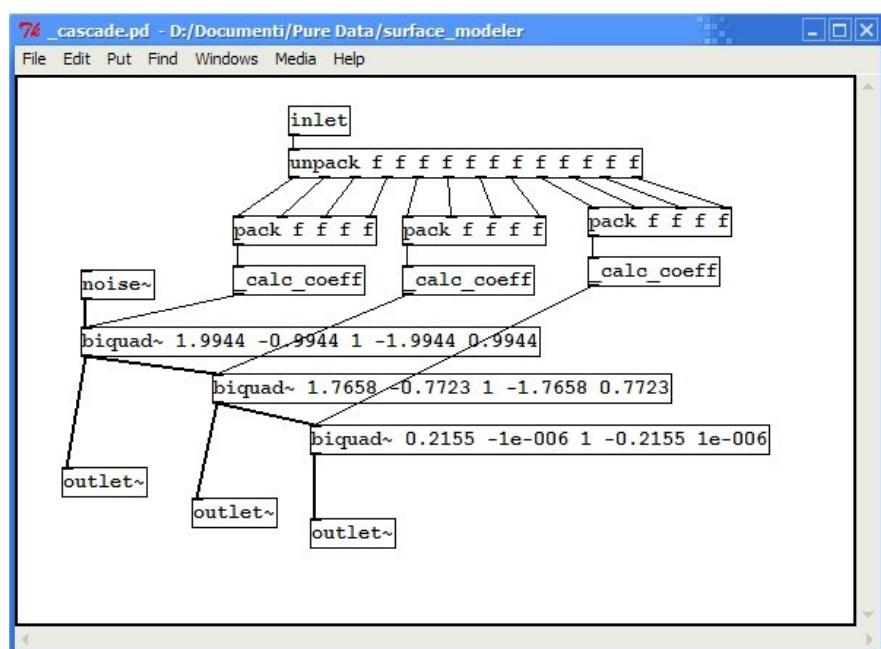
Come abbiamo visto è stato sviluppato un modello che descrive il suono per un corpo che rotola sopra una superficie. Se il corpo, invece di rotolare, striscia sulla superficie, provoca sempre la generazione di micro-contatti, che tuttavia avvengono con modalità diverse rispetto al rotolamento.

Una prima distinzione si ha nella velocità con cui avviene il contatto: se per il modello di rotolamento deve essere presa in considerazione la velocità angolare dell'oggetto che rotola (che nel caso di una sfera si calcola come  $\omega = v/r$ , con  $r$  raggio della sfera), per un corpo che striscia si deve considerare la velocità tangenziale, cioè la velocità lungo il piano sul quale giace la superficie. In secondo luogo il suono di un moto di rotolamento è spesso caratterizzato da irregolarità periodiche dovute alle caratteristiche particolari del corpo che rotola. Se questo non è perfettamente sferico e perfettamente liscio, i micro-contatti non saranno tutti uguali ma varieranno; in particolare per un corpo che rotola i micro-contatti si presentano con le stesse caratteristiche a scadenze periodiche (variabili con la velocità di movimento), e ciò si riflette nel suono prodotto, il quale sarà caratterizzato da variazioni periodiche. Per un corpo che striscia invece le irregolarità della sua superficie non comportano caratteristiche periodiche nel suono. Tutto ciò è valido se la dimensione del corpo è sufficientemente grande rispetto alla tessitura della superficie sulla quale si muove.

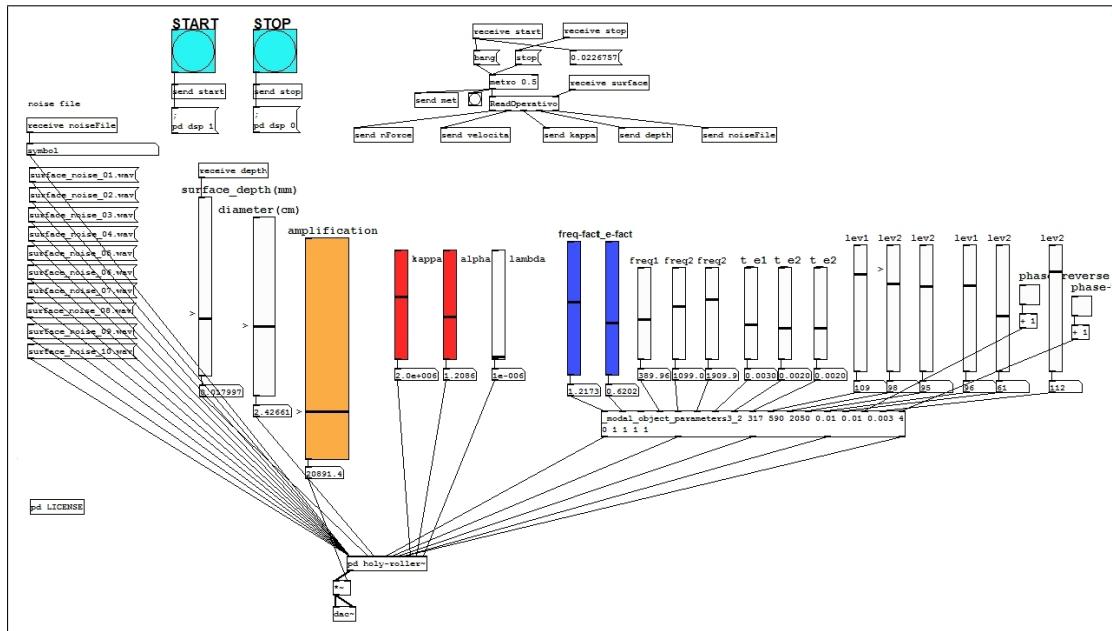
### 3.3. IMPLEMENTAZIONI DEI MODELLI IN PURE DATA



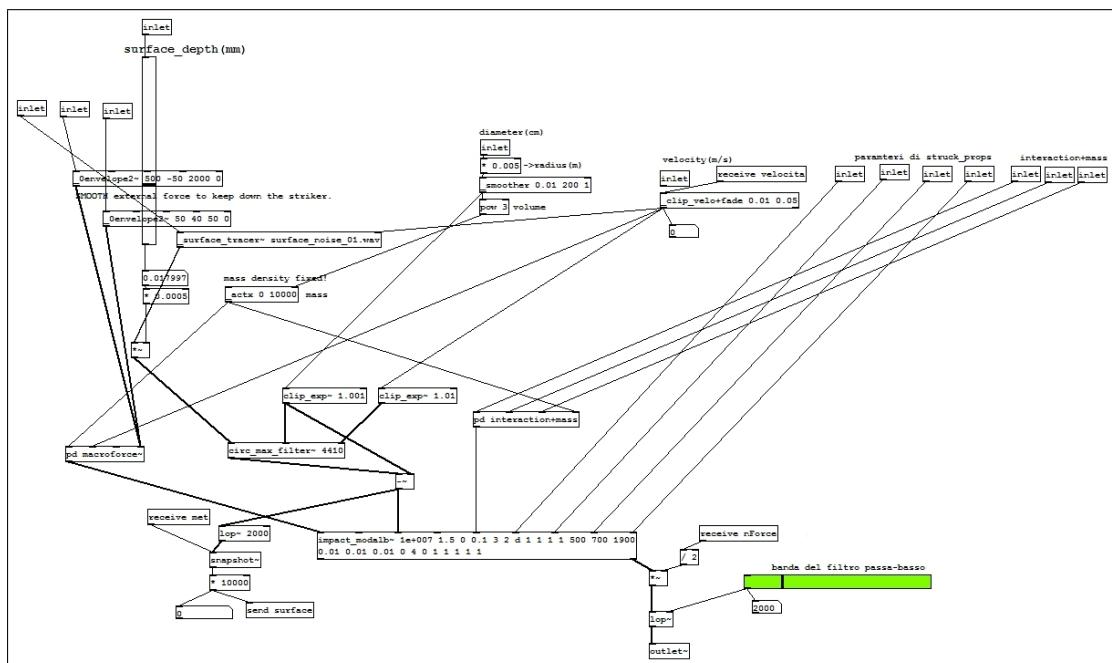
**Figura 3.11:** La subpatch `_initialize_fractal_noise`.



**Figura 3.12:** La subpatch `_cascade`.



**Figura 3.13:** La patch che implementa la generazione di rumore di frizione



**Figura 3.14:** La subpatch `holy_roller~`

La patch che implementa la generazione di rumore di sfregamento, mostrata in figura 3.13, è stata quindi elaborata a partire dalla patch che implementa il modello di rotolamento.

### **holy-roller~**

Il cuore della computazione viene svolto dalla subpatch **holy\_roller~** (figura 3.14); l'oggetto **holy\_roller~** possiede 13 inlet:

**inlet 0** – accetta un oggetto di tipo messaggio contenente il nome di un file .wav; tale file è stato precedentemente ottenuto registrando per circa 10 secondi l'output della patch generatrice di rumore frattale (e pertanto contiene a sua volta un rumore frattale);

**inlet 1 e 2** – non utilizzati in questa implementazione; accettano entrambi un segnale utilizzato poi come forza aggiuntiva da applicare all'oggetto rotolante;

**inlet 3** – accetta un numero in virgola mobile proporzionale all'amplificazione in ampiezza che deve subire il rumore frattale;

**inlet 4** – assumendo che l'oggetto che si muove sulla superficie sia una sfera, questo inlet riceve un numero in virgola mobile corrispondente al diametro della sfera in centimetri; sarà poi utilizzato per calcolarne la massa;

**inlet 5** – riceve un numero in virgola mobile indicante la velocità (in  $m/s$ ) dell'oggetto che si muove;

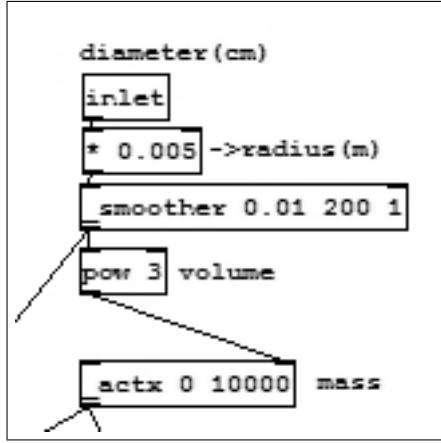
**inlet 6, 7, 8 e 9** – ricevono tutti dei segnali di controllo per l'impostazione delle frequenze e i tempi di decadimento degli oggetti modali usati nel modello di impatto (il tempo di decadimento è definito come il tempo richiesto affinché l'ampiezza decresca di un fattore  $1/e$  rispetto al suo valore iniziale);

**inlet 10** – riceve un numero in virgola mobile ( $k$ ) proporzionale alla rigidità dell'oggetto;

**inlet 11 e 12** – ricevono due numeri in virgola mobile ( $\alpha$  e  $\lambda$ ) utilizzati nel modulo **impact\_modalb~** per il calcolo della forza di impatto.

### **Elaborazione di diametro e velocità**

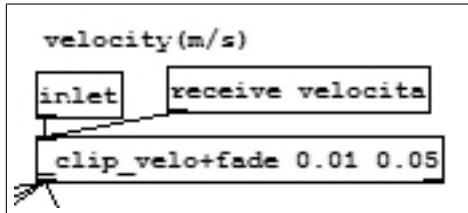
Il valore del diametro viene elaborato da alcuni blocchi allo scopo di calcolare dei valori indicanti il volume e la massa dell'oggetto rotolante. Il modulo



**Figura 3.15:** Blocchi di elaborazione del valore del diametro.

`_smoother` ha lo scopo di trasformare una variazione istantanea del valore del diametro secondo una rampa lineare della durata di un millisecondo.

Il valore di velocità invece viene inviato in input all'oggetto `_clip_velo+fade`; tale oggetto invia al primo outlet il valore di input se questo è maggiore del valore dato come primo argomento di costruzione (che nell'implementazione è 0.01), altrimenti viene inviato quest'ultimo. Il ruolo di questo outlet è impedire che alla patch successiva `clip_exp~` venga inviato costantemente un segnale di controllo nullo; se ciò si verificasse infatti si creerebbe un ciclo infinito che porterebbe ad un funzionamento non corretto della patch. Al secondo outlet viene inviato il valore ricevuto all'inlet opportunamente scalato nell'intervallo delimitato dai due argomenti.



**Figura 3.16:** L'oggetto `_clip_velo+fade`

### `clip_exp~`

I valori di raggio (in metri) e di velocità (in metri al secondo) vengono inviati ai due oggetti `clip_exp~`, la cui funzione è quella di limitare la variazione logaritmica dei segnali in input. Più precisamente, in ognuno dei due oggetti viene per prima cosa calcolato il rapporto tra due campioni in input a distanza di un millisecondo l'uno dall'altro; se tale rapporto è maggiore di `maxfact` o minore di `minfact` (calcolati a partire dal parametro di costruzione), il valore del campione corrente viene limitato e inviato all'outlet. Se il rapporto è minore del valore

predeterminato, il campione viene inviato in output invariato. Il codice seguente mostra come questo algoritmo sia implementato (per ragioni di efficienza) in C.

```
static t_int *clip_exp_tilde_perform(t_int *w)
{
    t_float *in = (t_float *) (w[1]);
    t_float *out = (t_float *) (w[2]);

    t_clip_exp_tilde_ctl *c = (t_clip_exp_tilde_ctl *) (w[3]);
    t_int buffersize = (t_int) (w[4]);

    t_float input, ratio;

    // esamina tutto il buffer
    while (buffersize--)
    {
        input = *in++;

        // se last è diverso da zero
        if (c->last != 0.)
        {
            // se ratio è compreso tra maxfact e minfact
            // in ouput viene mandato input
            // altrimenti l'output è impostato a maxfact o minfact
            ratio = input / c->last;

            if (ratio > c->maxfact)
                c->last *= c->maxfact;

            else if (ratio < c->minfact)
                c->last *= c->minfact;

            else
                c->last = input;
        }

        *out++ = c->last;
    }

    return (w+5);
}
```

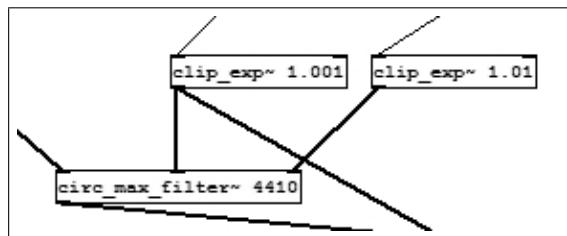
I valori `maxfact` e `minfact` vengono calcolati nel seguente modo:

```
static void set_expmax(t_clip_exp_tilde *x, t_floatarg expmax)
{
    t_clip_exp_tilde_ctl *c = x->x_ctl;

    // se l'argomento del modulo è < 1
    // pongo maxfact e minfact = 1
    if (expmax <= 1.)
    {
        c->maxfact = c->minfact = 1.;
        post("clip_exp: expmax <= 1?! Is set to 1.");
    }
    // altrimenti:
    // maxfact = expmax^(1000/samprate)
    else
    {
        c->maxfact = pow(expmax, 1000. / x->samprate);
        c->minfact = 1. / c->maxfact;
    }
}
```

### circ\_max\_filter~

Gli outlet dei due moduli `clip_exp~` sono collegati al secondo e terzo inlet dell'oggetto `circ_max_filter~`: la sua funzione è quella di tracciare il profilo



**Figura 3.17:** Gli oggetti `clip_exp~` e `circ_max_filter~`.

della superficie sulla quale l'oggetto rotola e calcolare i punti di contatto tra i due. Nel primo inlet entra il controllo di segnale ottenuto dalla moltiplicazione del rumore frattale per il fattore di amplificazione `surface_depth`. Il ciclo principale svolto dal modulo è il seguente:

```
static t_int *circ_max_filter_perform(t_int *w)
{
```

```

t_float *in1 = (t_float *) (w[1]);
t_float *in2 = (t_float *) (w[2]);
t_float *in3 = (t_float *) (w[3]);
t_float *out = (t_float *) (w[4]);

t_circ_max_filter_ctl *c = (t_circ_max_filter_ctl *) (w[5]);
t_float *p_samrate = (t_float *) (w[6]);
t_int buffersize = (t_int) (w[7]);

t_float input, radius, velocity;
t_int range;

while (buffersize--)
{
    input = *in1++;
    radius = *in2++;
    velocity = *in3++;

    inc_bottom_ivalue1_circ_buff_1float2int(c->p_peaks, -1);

    while ((range = bottom_ivalue1_circ_buff_1float2int(c->p_peaks))
        < bottom_ivalue2_circ_buff_1float2int(c->p_peaks))
    {
        delete_bottom_circ_buff_1float2int(c->p_peaks);
        inc_bottom_ivalue1_circ_buff_1float2int(c->p_peaks, range);
    }

    *out++ = up_circle(velocity * range / *p_samrate, radius)
        + bottom_fvalue_circ_buff_1float2int(c->p_peaks);

    to_buffer(c->p_peaks, *p_samrate, input, radius, velocity, 1);
}

return (w+8);
}

```

In particolare si può notare come l'istruzione

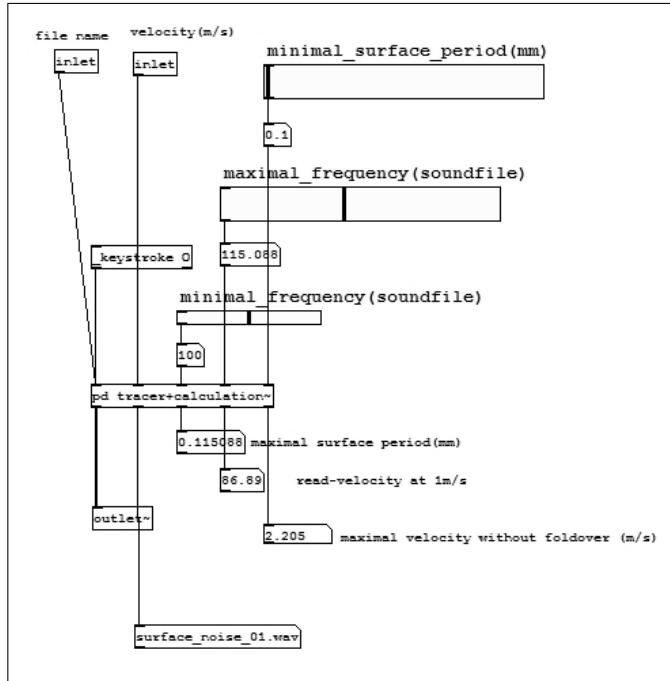
```

*out++ = up_circle(velocity * range / *p_samrate, radius)
    + bottom_fvalue_circ_buff_1float2int(c->p_peaks);

```

calcola i punti di contatto svolgendo la computazione in 3.2.

La funzione `up_circle` richiede due argomenti; dopo aver calcolato i quadrati di questi, ritorna la radice quadrata della differenza dei due:



**Figura 3.18:** La subpatch `_surface_tracer~`

```
static INLINE t_float up_circle(t_float x, t_float radius)
{
    t_float x_2 = x*x, radius_2 = radius*radius;

    if (x_2 >= radius_2)
    {
        return(0.);
    }
    else
        return(sqrt(radius_2 - x_2));
}
```

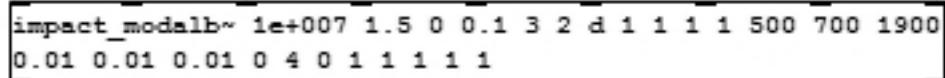
La funzione `to_buffer` si occupa di aggiornare il profilo della superficie in base al segnale ricevuto al primo inlet.

### `_surface_tracer~`

Il file con estensione `.wav` contenente il rumore frattale viene letto dalla subpatch `_surface_tracer~` (figura 3.18). Questa subpatch, assieme alle subpatch in essa contenute quali `pd tracer+calculation~`, `soundfiler_tracer~` e `table_tracer~`, legge il file audio e lo scrive in un array; successivamente, per inviare i campioni in output, esegue una ricerca di tipo *table look-up* con frequenza dipendente dalla velocità di movimento dell'oggetto sulla superficie.

### impact\_modalb~

Le successive elaborazioni dei segnali finora calcolati sono svolte dal modulo **impact\_modalb~**, un oggetto che implementa il modello descritto in [1] per i suoni di impatto.



```
impact_modalb~ 1e+007 1.5 0 0.1 3 2 d 1 1 1 1 500 700 1900
0.01 0.01 0.01 0 4 0 1 1 1 1 1
```

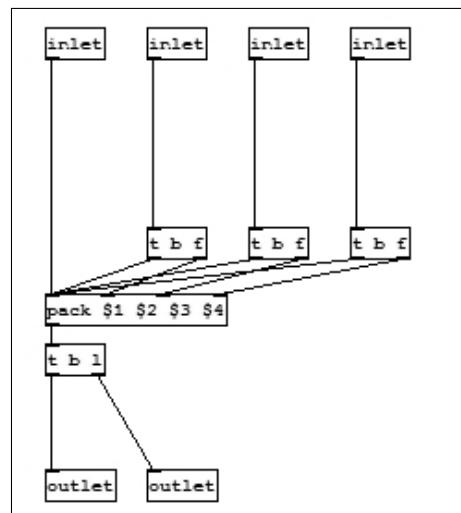
**Figura 3.19:** L'oggetto `impact_modalb~`

In questa implementazione vengono utilizzati due modi e tre punti di interazione.

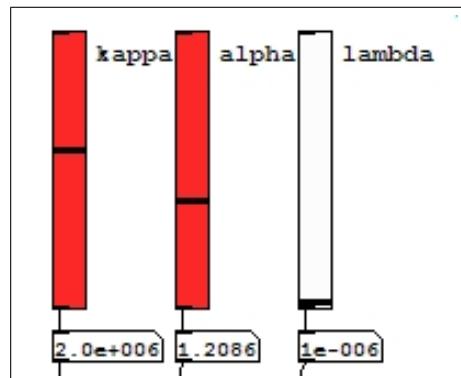
I parametri della forza di contatto e della massa vengono ricevuti dalla subpatch `interaction+mass` e la subpatch in essa contenuta `_par_to_list4` (figura 3.20); in output (secondo outlet) viene mandata una lista contenente queste informazioni. Nei quattro inlet vengono ricevuti, in ordine:  $k$ ,  $\alpha$ ,  $\lambda$  e la massa del percussore.

La subpatch `_modal_object_parameters3_2`, dove 3 è il numero di modi e 2 il numero di punti di interazione, raccoglie i parametri del risonatore. Nei primi tre inlet entrano i fattori moltiplicativi per frequenza, tempo di decadimento e guadagno; ai successivi tre inlet sono collegati i controlli delle frequenze di tutti i modi, poi i tempi di decadimento di tutti i modi. Gli ultimi inlet ricevono i livelli di ciascun modo per ogni punto di interazione con l'eventuale possibilità di invertire la fase (*phase-reverse*). In uscita sono presenti cinque outlet: il primo per la lista dei fattori, il secondo per la lista delle frequenze, il terzo per la lista dei tempi di decadimento e un outlet per ogni punto di interazione con l'indice del punto di interazione seguito dalla lista dei livelli (con l'eventuale fattore di *phase-reverse*). Infine l'oggetto `_modal_object_parameters3_2` deve essere inizializzato con la seguente lista di argomenti: lista dei valori delle frequenze, lista dei valori dei tempi di decadimento, valori dei punti di interazione e del *phase-reverse*; un valore 1 per il livello corrisponde ad una impostazione del relativo slider a 100, dato che quest'ultimo viene convertito in dB RMS. L'oggetto `impact_modalb~` possiede i seguenti argomenti di costruzione:

- valori di default di  $k$ ,  $\alpha$ ,  $\lambda$  e massa del percussore;
- numero di modi e numero di punti di interazione;
- maschera dei punti di interazione;
- valori di default dei tre fattori di guadagno;

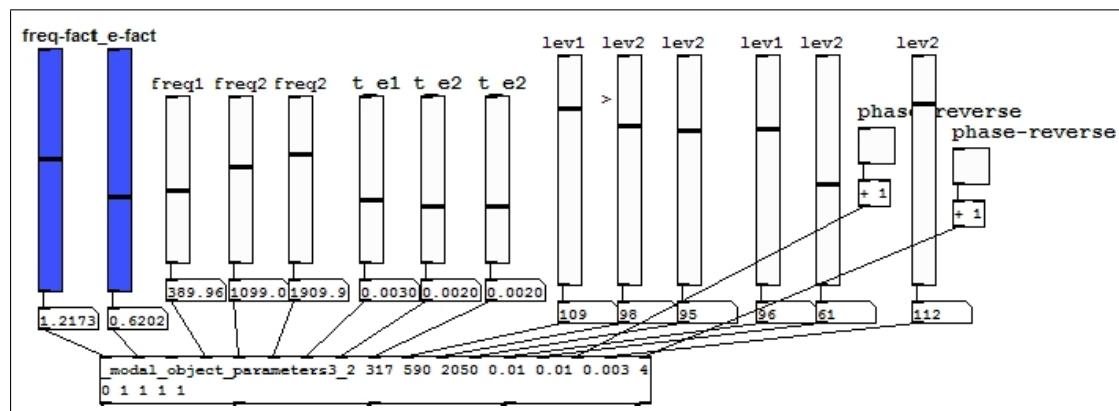


(a)



(b)

**Figura 3.20:** La subpatch `_par_to_list4`(a) e i parametri in ingresso (b).



**Figura 3.21:** La subpatch `_modal_object_parameters3_2` e i parametri del risonatore.

- valori di default delle frequenze;
- valori di default dei tempi di decadimento;
- per ogni punto di interazione il suo indice (partendo da 0) seguito dai valori dei livelli.

```
impact_modalb~ 1e+007 1.5 0 0.1|3 2|d 1|1 1 1|500 700 1900
0.01 0.01 0.01|0 4 0 1|1 1 1 1
```

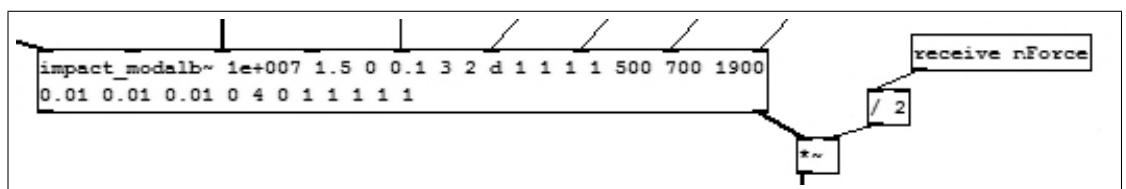
**Figura 3.22:** L’oggetto impact\_modalb~ e la divisione tra i gruppi di argomenti di costruzione.

### Dipendenza dell’ampiezza del suono dalla forza normale

Secondo studi svolti da Van Den Doel, Kry e Pai [46], nei contatti che avvengono tra due corpi e che coinvolgono forze di frizione, queste ultime sono calcolabili come:

$$F_{frizione} = \mu F_{normale} \quad (3.12)$$

e il volume del suono prodotto da ogni contatto è proporzionale a  $\sqrt{v \cdot F_{normale}}$ , dove  $v$  è la velocità alla quale avviene il contatto; in questo calcolo si assume che l’energia acustica sia proporzionale alla perdita di capacità da parte della superficie di opporre una resistenza (di frizione) al moto. Nella subpatch holy\_roller~ tale caratteristica è implementata tramite gli oggetti in figura 3.23.



**Figura 3.23:** Implementazione dell’amplificazione proporzionale alla radice quadrata del prodotto tra velocità e forza normale.



# Capitolo 4

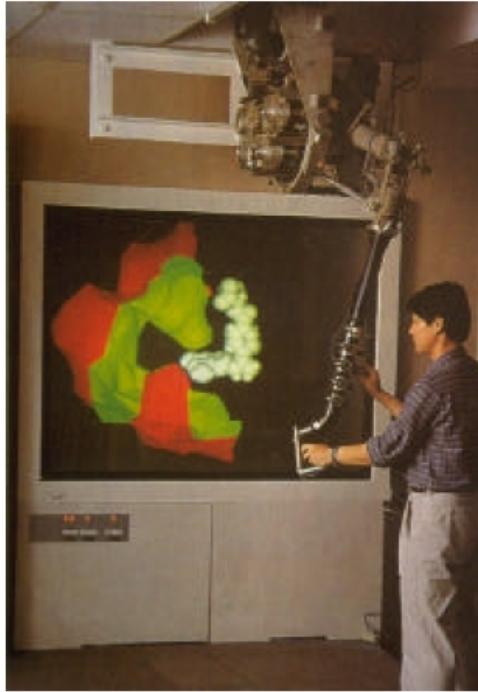
## I dispositivi aptici

### 4.1 Cosa sono i dispositivi aptici

Negli anni settanta e ottanta la ricerca nel campo della robotica ha iniziato a dirigere la sua attenzione verso la percezione e la manipolazione tattile per la creazione di robot autonomi; negli anni novanta invece la presenza di nuove tecnologie ha permesso la nascita della *computer haptics*, cioè una virtualizzazione che, come nella computer graphics, permette di simulare oggetti in maniera interattiva. Sebbene Knoll abbia dimostrato già negli anni sessanta la possibilità di eseguire una interazione aptica con semplici oggetti, solo le tecnologie recenti hanno reso possibile la realizzazione di tale interazione con oggetti sempre più complessi, combinando insieme dispositivi ad alte prestazioni, modelli di calcolo geometrici, tecniche di rilevamento delle collisioni e comprensione del sistema tattile umano.

I dispositivi aptici si comportano come piccoli robot che effettuano uno scambio di energia con l’utente; in particolare permettono all’utente di interagire con un ambiente virtuale ricevendo un feedback tattile, ottenuto applicando delle forze opposte al movimento dell’utente lungo i tre assi  $x$ ,  $y$  e  $z$ . La caratteristica principale che li distingue dalle altre tipologie di dispositivi è la loro *bidirezionalità*, cioè la possibilità di ricevere informazioni dall’utente e di inviare informazioni all’utente. Infatti, durante il normale utilizzo di un tale dispositivo, l’uomo imprime una forza al meccanismo aptico (uno stilo, un ditale o un altro componente diverso a seconda del tipo di interfaccia) e ne cambia la posizione; successivamente è il dispositivo a riflettere una forza calcolata in base al movimento al quale è stato sottoposto.

I *Bilateral Master–Slave Manipulator* (MSMs) fanno parte dei primi dispositivi aptici sviluppati e venivano impiegati nell’industria nucleare per manipolare in modo sicuro e remoto il materiale irradiato. Il termine MSMs deriva dal fatto



**Figura 4.1:** Uso di un braccio meccanico per la manipolazione di molecole in un ambiente di realtà virtuale.

che il braccio meccanico *master* è tipicamente una riproduzione di un braccio remoto *slave*, e i due sono legati da catene, cavi o altri sistemi elettromeccanici. Un altro dei primi usi dei bracci meccanici master è avvenuto nell'ambito della realtà virtuale per la manipolazione delle molecole e l'interazione aptica con la simulazione della forza eletrostatica molecola–substrato; un esempio di questo è in figura 4.1 [45]. Altri progetti importanti sono stati sviluppati nel campo del force feedback per la chirurgia, come il progetto IERAPSI dello European Union Framework, un ambiente per le prove e la pianificazione degli interventi chirurgici (figura 4.2), oppure nel campo militare per il training sul rilevamento e disinnesco delle mine.

Il termine *aptico* (dal greco *haptesthai* che significa *toccare*) indica la parte di fisiologia inherente il senso del tatto, mentre con *feedback aptico* (haptic feedback) si indica sia la consapevolezza degli stimoli ricevuti in conseguenza di un contatto con una superficie, sia le informazioni cinetiche ricevute in base alla posizione e al movimento del corpo. Lo scambio bidirezionale di informazioni con un dispositivo aptico viene chiamato *percezione aptica* (haptic perception).

In figura 4.3 è riportato lo schema di un'interazione aptica; nella parte sinistra si vede come avvengono le elaborazioni all'interno del corpo umano: i recettori posti sulla pelle ricevono le informazioni sulla forza e le trasmettono al cervello, il quale comanda poi i muscoli del braccio e della mano. Il dispositivo invece (parte destra) rileva il moto attraverso dei sensori che inviano i dati sulla posi-



**Figura 4.2:** Sistema di feedback aptico usato per la pianificazione di interventi chirurgici.

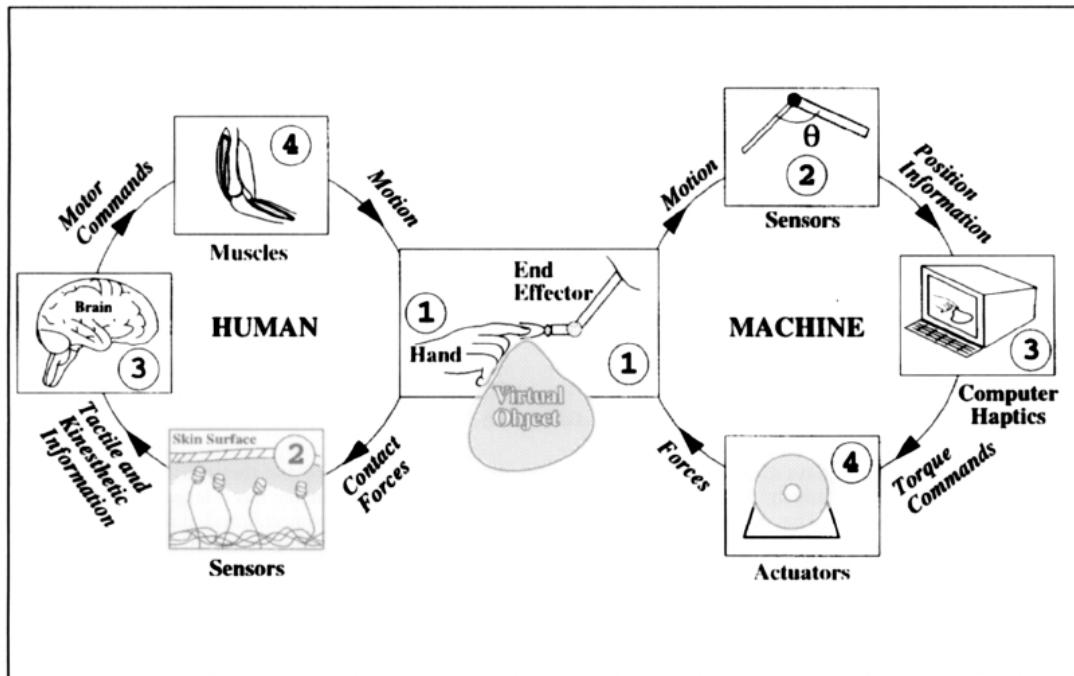
zione al computer al quale il dispositivo stesso è collegato; quest'ultimo elabora i dati ricevuti e invia i comandi di torsione agli attuatori che generano la forza di resistenza nel dispositivo.

## 4.2 Tipi di dispositivi

I dispositivi aptici possono essere distinti in base al loro posizionamento:

- i dispositivi *ground based* sono quelli che vengono appoggiati ad un tavolo di lavoro e includono i joystick che riflettono le forze e le interfacce aptiche desktop, come il Phantom® Omni™ o il Phantom® Desktop™ (figura 4.4) di SensAble Technologies;
- gli *esoscheletri meccanici* sono dei dispositivi che vengono indossati dall'utente su braccia o gambe (figura 4.5);
- i *guanti force feedback* leggono le informazioni di contatto delle singole dita e restituiscono le forze di resistenza, ma non possono riprodurre sensazioni di oggetti pesanti o inerzie (figura 4.6).

Un altro tipo di distinzione è tra dispositivi a *impedenza* o *ammettenza*: i primi leggono informazioni relative alla posizione e inviano in output informazioni sulle forze, i secondi leggono le forze e inviano i dati sulla posizione. Il terzo modo di classificare i dispositivi è in base ai gradi di libertà che caratterizzano il loro movimento:



**Figura 4.3:** Interazione aptica tra utente e dispositivo.

**Dispositivi ad un grado di libertà** – Misurano e applicano le forze lungo una sola direzione. Esempi di movimenti reali ad un grado di libertà includono l'apertura di una porta, l'uso di un paio di forbici o la pressione del pistone di una siringa. La simulazione aptica ad un grado di libertà più comune è il *muro virtuale*, che corrisponde al rendering di una forza che si manifesta al contatto con una superficie molto rigida.

**Dispositivi a due gradi di libertà** – Il mouse è un semplice esempio di dispositivo a due gradi di libertà; possono essere utilizzati per interagire con oggetti tridimensionali mappando opportunamente i punti di contatto a tre gradi di libertà su un piano.

**Dispositivi a tre gradi di libertà** – In questo caso la mappatura dallo spazio tridimensionale del dispositivo a quello virtuale è più immediata, in quanto ogni componente della posizione nel sistema di riferimento solidale al dispositivo deve essere trasformata nella corrispondente componente nel sistema di riferimento virtuale. Nell'interazione con gli oggetti virtuali il modulo della forza è calcolato in base a quanto il dispositivo penetra l'oggetto, mentre per il calcolo della direzione sono stati sviluppati vari metodi, tra i quali quello maggiormente utilizzato è l'algoritmo del *proxy* (illustrato nel paragrafo 4.3).



**Figura 4.4:** Dispositivo aptico *Phantom® Desktop™* di *SensAble Technologies*.

**Dispositivi a più di tre gradi di libertà** – Per rendere più realistici gli scenari virtuali si è reso necessario introdurre la possibilità di effettuare torsioni. Barbagli [2] ha scritto un algoritmo che supporta l’interazione con un punto di contatto con frizione e momenti per simulare quattro gradi di libertà; la simulazione di cinque gradi di libertà (come nel caso del contatto tra un segmento e un oggetto) è stata implementata da Basdogan [3], mentre studi sulle interazioni a sei gradi di libertà sono stati eseguiti da McNeely [31] e Otaduy e Lin [34].

### Caratteristiche e prestazioni dei dispositivi

Le prestazioni dei vari tipi di dispositivi dipendono dalle abilità e limitazioni umane e dell’utente; le simulazioni di oggetti e ambienti virtuali sono sempre basate su calcoli approssimati, e sono proprio le limitazioni della sensibilità dell’utente a determinare se tali approssimazioni sono sufficienti. Le caratteristiche che un’interfaccia aptica dovrebbe possedere sono [43]:

- Livelli di inerzia e frizione bassi, assieme ad assenza di costrizioni cinematiche imposte dal dispositivo, in modo tale che il movimento libero sia av-



**Figura 4.5:** Dispositivo aptico CyberForce™ composto da un esoscheletro poggiato direttamente al suolo, [www.immersion.com](http://www.immersion.com).

vertito effettivamente come tale (in altre parole, l’utente non deve avvertire forze di resistenza quando queste non vengono simulate dall’applicazione).

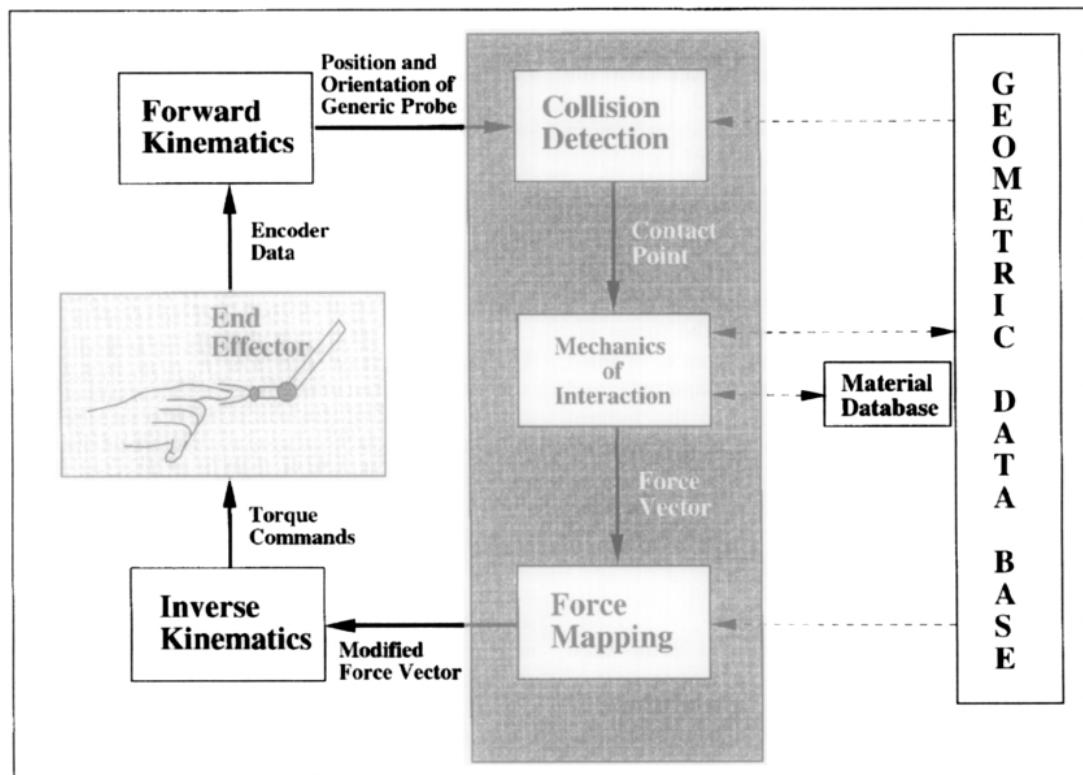
- La risoluzione, sia in termini di posizione che di forza riflessa, deve corrispondere a quella del sistema tattile umano e dipende dal processo nel quale viene impiegato il dispositivo. In particolare l’utente:
  - non deve poter attraversare gli oggetti applicando una forza eccessiva;
  - non deve avvertire vibrazioni non volute (come quelle dovute a quantizzazioni);
  - non deve avvertire come elastici gli oggetti che invece sono rigidi e viceversa.
- Ergonomia e comfort: i dispositivi non devono arrecare disturbi all’utente che li indossa o li utilizza, in quanto tali disturbi potrebbero sovrastare tutte le altre sensazioni, in particolare quelle aptiche.

### 4.3 Struttura di una pipeline di rendering aptico

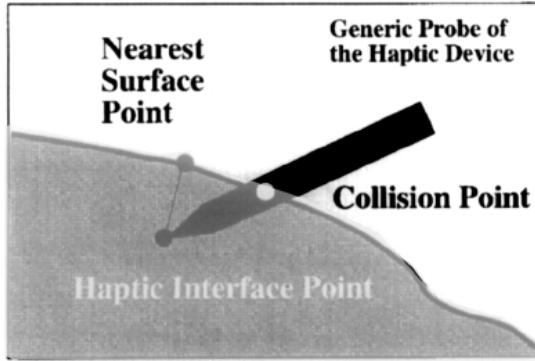
La struttura di una pipeline di rendering aptico è relativamente semplice ed è illustrata in figura 4.7. Quando l’utente muove la sonda del dispositivo, i nuovi valori di posizione e orientazione vengono catturati dai codificatori; successivamente vengono rilevate le eventuali collisioni con gli oggetti virtuali: se la sonda entra in contatto con un oggetto, viene calcolata la forza di reazione in base alla profondità della penetrazione della sonda nell’oggetto. La forza calcolata viene



**Figura 4.6:** I guanti (esoscheletri) aptici CyberGrasp<sup>TM</sup>(a) e CyberGlove<sup>TM</sup>(b), [www.immersion.com](http://www.immersion.com).



**Figura 4.7:** Processo associato al rendering delle forze. Le linee in grassetto rappresentano i flussi del processo, mentre quelle tratteggiate rappresentano lo scambio di informazioni.



**Figura 4.8:** Interazioni aptiche point-based e ray-based.

poi mappata sulla superficie dell'oggetto in modo da tenere in considerazione i dettagli di quest'ultimo; il vettore delle forze così modificato viene inviato al dispositivo e, tramite questo, all'utente.

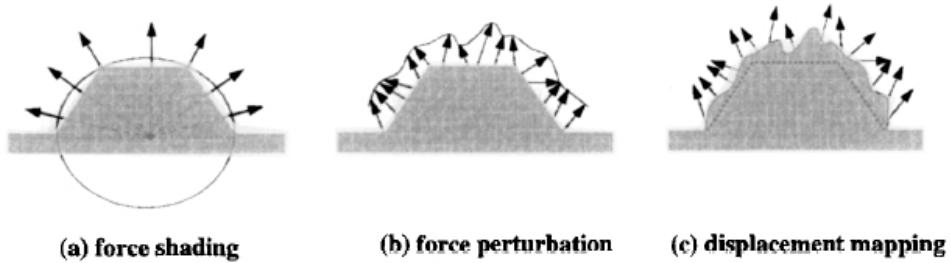
Le diverse tecniche di rendering possono essere distinte a seconda del tipo di interazione in *point based* e *ray based* (figura 4.8):

- Con l'interazione point-based solo l'estremità del dispositivo (*HIP*, haptic interface point) entra in contatto con gli oggetti. Dato che questi hanno una rigidità finita, il punto HIP li penetra e la profondità della penetrazione è calcolata come la minima distanza tra HIP e superficie dell'oggetto virtuale.
- Nell'interazione ray-based la sonda è modellata come un raggio finito; l'algoritmo di collisione individua come punto di contatto l'intersezione tra il raggio e la superficie dell'oggetto, mentre come profondità della penetrazione si considera la distanza tra HIP e punto di collisione lungo la normale alla superficie.

In entrambi i casi la forza è calcolata usando la legge di Hooke  $F = kx$ , dove  $x$  è la stessa direzione lungo la quale viene calcolata la profondità di penetrazione; se le interazioni sono prive di forze di frizione, la forza di reazione è normale alla superficie nel punto di contatto.

Zilles e Salisbury [49] hanno sviluppato un algoritmo più sofisticato per il rendering delle forze basato sulla tecnica point-based; hanno definito un nuovo punto, detto *god-object* o *proxy*, rappresentante la locazione sulla superficie del punto di contatto che viene calcolato ogni volta che il dispositivo viene mosso, con il vincolo ulteriore che la distanza tra god-object e HIP deve essere minimizzata, mentre il god-object resta sempre sulla superficie dell'oggetto anche quando il punto HIP lo penetra.

Un approccio più semplice consiste nel calcolare un'approssimazione della superficie come un piano tangente al punto di contatto; tale piano viene aggiornato



**Figura 4.9:** Tecniche di rendering aptico dei dettagli di una superficie; le frecce rappresentano i vettori delle forze riflesse. L'area in grigio rappresenta la geometria dell'oggetto mentre la linea nera indica la geometria della superficie come viene percepita dall'utente.

ad una frequenza inferiore a quella propria del rendering delle forze, pertanto può portare a sentire delle discontinuità se il dispositivo viene mosso su grandi distanze prima che il piano venga aggiornato.

Un'altra tecnica consiste nel considerare gli oggetti virtuali come volumi formati da *voxel* (elementi di volume rappresentanti un valore su una griglia regolare nello spazio tridimensionale); ad ogni voxel vengono associati otto byte di informazioni, includenti i valori di densità del materiale, gradiente di densità, colore e altre proprietà aptiche come viscosità ed elasticità. Quando il punto HIP si trova all'interno dell'oggetto, il valore scalare di densità al punto di contatto viene usato per calcolare la forza di reazione attraverso delle trasformazioni lineari, mentre il valore di gradiente della densità viene usato per determinare la direzione normale alla superficie.

Basdogan ha sviluppato una tecnica di rendering ray-based [3] nella quale le coordinate delle due estremità della sonda vengono aggiornate ad ogni movimento, rilevando ogni collisione tra il raggio e l'oggetto virtuale e calcolando la forza secondo la legge di Hooke. Il rilevamento delle collisioni avviene in tre passi: per prima cosa viene rilevata la collisione tra il raggio e lo spazio rettangolare contenente l'oggetto virtuale, poi tra il raggio e lo spazio che circonda un qualsiasi elemento triangolare; infine viene rilevato il contatto tra il raggio e l'elemento triangolare usando calcoli geometrici. La divisione in più passi porta ad un aumento di prestazioni consentendo di lavorare a frame rate più elevati.

#### 4.4 Rendering dei dettagli della superficie

Esistono diverse tecniche per effettuare il rendering dei dettagli della superficie, diverse a seconda del tipo di sensazione che deve essere data: superficie smussata, superficie con tessitura o superficie con frizione.

#### 4.4.1 Rendering di superfici smussate

Quando gli oggetti vengono rappresentati tramite insiemi di poligoni, l'utente non percepisce la forma che si intendeva rappresentare, ma avverte i lati e gli angoli dei vari poligoni. Per minimizzare tale effetto Morgenbesser e Srinivasan [44] hanno elaborato una tecnica chiamata *force shading*: con tale metodo il vettore della forza viene interpolato lungo la superficie in modo che la sua direzione vari con continuità (figura 4.9(a)), di conseguenza l'utente avverte una superficie più smussata rispetto alla sua rappresentazione originale. Diventa così possibile sviluppare modelli geometrici di diverso livello di dettaglio ed effettuare poi il rendering con il modello di forza dettagliato o con il modello force–shaded a seconda dei requisiti dell'applicazione.

Il force shading potrebbe diventare poco efficiente quando gli angoli tra i poligoni che rappresentano una superficie sono al di sotto di un certo valore; in questo caso è sufficiente diminuire il numero di poligoni utilizzati fino a quando gli angoli tra questi sono tutti maggiori del valore critico.

#### 4.4.2 Rendering delle tessiture aptiche

Srinivasan e Basdogan [43] hanno sviluppato due metodologie per il rendering delle tessiture di superfici tridimensionali che prendono spunto da quelle usate nella computer graphics:

**Perturbazione delle forze** – La perturbazione delle forze consiste nel modificare la direzione e il modulo del vettore delle forze per generare effetti sulla superficie (come la ruvidità). Max e Becker [30] hanno sviluppato una formula che permette di generare delle ruvidità visuali, basandosi sulla normale alla superficie originale:

$$M = N - \nabla h + (\nabla h \cdot N)N \quad (4.1)$$

dove  $M$  è la normale modificata,  $N$  è la normale originale e  $\nabla h$  è il gradiente della profondità della tessitura. Lo stesso metodo può essere usato per generare la ruvidità aptica (figura 4.9(b)).

**Displacement mapping** – In questo caso, invece di modificare il vettore delle forze, viene modificata la geometria della superficie (figura 4.9(c)). Per generare micro–tessiture è necessario che la superficie sia composta da un numero elevato di poligoni, in modo tale da rendere possibile la sua modifica punto per punto; tutto ciò però incrementa considerevolmente il carico computazionale. Usare questa tecnica per il rendering aptico comporta ulteriori problemi: gli oggetti virtuali non possono essere infinitamente rigidi,

perciò il dispositivo penetra all'interno della superficie; oppure il rilevamento delle collisioni o la localizzazione dei punti sulla superficie ogni volta che la sonda viene mossa crea ambiguità dovute all'esistenza di micro-dettagli, portando a delle discontinuità delle forze. Si capisce come il displacement mapping sia più indicato per il rendering di macro-tessiture.

Altre tecniche invece si basano sull'uso della frequenza e della profondità delle tessiture:

**Tessiture aptiche basate sulle immagini** – L'idea consiste nell'utilizzare le informazioni dell'immagine bidimensionale usata come texture dell'oggetto virtuale; assumendo che le intensità di grigio dell'immagine possano essere usate direttamente come indicatori della profondità della tessitura aptica, possiamo associare ogni coordinata della texture  $(u, v)$  alle coordinate di ogni vertice  $(x, y, z)$ . Viene poi calcolato il gradiente della profondità al punto di collisione e il vettore della forza è perturbato di conseguenza.

**Tessiture aptiche procedurali** – L'obiettivo è generare le tessiture aptiche mediante funzioni matematiche, usandole poi per modificare il gradiente  $\nabla h$  e la forza di reazione al punto di contatto, o per modificare la geometria dell'oggetto; un esempio è la simulazione di tessiture stocastiche [10].

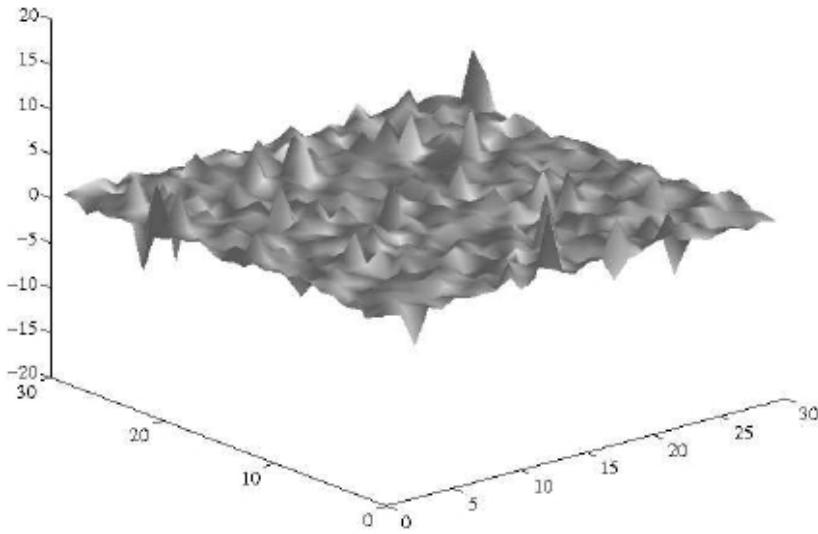
#### 4.4.3 Rendering delle superfici con frizione

L'aggiunta di forze di frizione rende la simulazione aptica più realistica; ad esempio, senza frizione non saremmo in grado di premere un pulsante virtuale in quanto il dispositivo scivolerebbe su di esso.

Molti ricercatori hanno proposto l'utilizzo di frizioni di Coulomb o frizioni viscose; le prime hanno una componente di frizione statica e una di frizione dinamica, mentre le altre sono dipendenti dalla velocità. Entrambi i tipi esercitano una forza opposta e tangenziale al punto di contatto. Se la misurazione della velocità di contatto è affetta da rumore o i cambiamenti di posizione sono troppo veloci, l'interazione aptica potrebbe diventare instabile.

### 4.5 Generazione di tessiture casuali con il metodo stocastico

Definiamo *spazio di tessitura* un volume di tessitura rappresentato da vettori tridimensionali; un campione in un punto qualsiasi è un vettore tridimensionale senza vincoli su modulo o direzione. Questo spazio continuo viene generato tramite vari processi che possono essere deterministici, stocastici o misti.



**Figura 4.10:** Texture bidimensionale composta da due pdf gaussiane.

Le caratteristiche di una tessitura possono essere descritte tramite il suo spettro di potenza, e variando lo spettro si ottengono diverse tessiture.

#### 4.5.1 Modelli stocastici

I modelli stocastici sono caratterizzati da misure statistiche di una immagine di texture; spesso sono sufficienti le misure di basso ordine. Il pattern della texture viene sintetizzato applicando ad un rumore bianco un filtro basato sulla funzione di autocorrelazione.

Dato che l'obiettivo non è simulare tessiture reali, bensì tessiture che siano percettibilmente diverse l'una dall'altra, è possibile generare le texture proprio processando un rumore tramite un filtro. Il rumore deve essere a banda limitata (rumore rosa), stazionario (invariante rispetto alla traslazione), isotropico e non deve essere periodico; il rumore bianco filtrato possiede queste proprietà. In particolare si considera il *rumore bianco gaussiano*: una trasformazione di un vettore casuale gaussiano resta gaussiano ed è completamente definito dalle sue statistiche di primo e secondo ordine. Semplicemente modificando la varianza (e di conseguenza lo spettro di potenza) è possibile ottenere tessiture che vanno da lisce a ruvide [42].

Per tessiture complesse si può utilizzare una *pdf* (probability density function) ottenuta da varie pdf gaussiane:

$$f(x) = \sum_{i=1}^M a_i N(\mu_i, C_i), \quad (4.2)$$

dove  $\sum_{i=1}^M a_i = 1$  e  $N(\cdot)$  è l'i-esima pdf gaussiana con media  $\mu_i$  e matrice di covarianza  $C_i$ ; i termini  $a_i$  possono essere variati arbitrariamente, o modellati in base a tessiture reali. In figura 4.10 è rappresentata una texture bidimensionale composta da due pdf gaussiane.

Tutte queste metodologie producono campioni indipendenti e identicamente distribuiti, ottenendo così uno spettro di potenza costante; per modificare tale spettro è sufficiente filtrarlo con tecniche che affiancano trasformazioni statistiche al normale filtraggio. Un esempio è costituito dal moto Browniano visto nel paragrafo 3.2, un processo invariante rispetto alla scala descritto da un parametro di auto-similarità  $h$  e uno spettro di potenza di tipo  $1/f$  [8, 16].

### 4.5.2 Procedure implicite e sintesi spettrale di Fourier

La creazione di uno spettro di potenza è possibile anche tramite procedure implicite, usando ad esempio delle sinusoidi, che nel dominio della frequenza diventano degli impulsi. Dalla teoria di Fourier si sa che ogni segnale periodico può essere visto come somma di infinite sinusoidi: così la creazione di una texture diventa un procedimento di sintesi spettrale di Fourier. Le caratteristiche dello spettro vengono modificate modulando l'ampiezza e la frequenza delle sinusoidi, mentre altri effetti spaziali possono essere ottenuti con variazioni di fase. Possono essere utilizzate anche altre funzioni, come dente di sega o onde quadre, le quali aggiungono componenti in alta frequenza ad intervalli armonici; segnali di questo tipo diventano delle vibrazioni in un'interfaccia aptica, fatto che può rendere più realistica la tessitura.

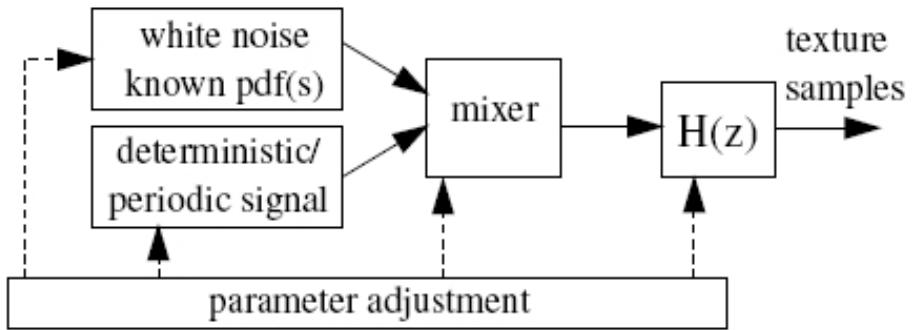
Se il periodo dei segnali è sufficientemente ampio, le funzioni periodiche possono essere usate per creare tessiture che appaiono casuali. Un altro modo per creare tessiture casuali con funzioni periodiche è l'uso di un processo stocastico: il vettore delle forze della tessitura  $F_t$  alla posizione  $p$  può essere espresso tramite la seguente equazione:

$$F_t(p) = Sg(p) + N, \quad (4.3)$$

dove  $g(p)$  è una funzione implicita e deterministica,  $S$  è un rumore moltiplicativo e  $N$  rappresenta un rumore sovrapposto. Il rumore moltiplicativo può essere usato per scalare casualmente una funzione deterministica apportando ulteriori variazioni alla tessitura.

### 4.5.3 Filtraggio

Il filtraggio viene usato per modellare lo spettro di potenza allo scopo di ottenere una tessitura con determinate caratteristiche. Una classificazione qualitativa



**Figura 4.11:** Schema a blocchi per modellare una texture tramite filtraggio;  $H(z)$  è la funzione di trasferimento complessiva.

delle tessiture può così essere tradotta in una descrizione spettrale. Ad esempio, le tessiture possono essere ruvide, lisce, fini, granulate, regolari o irregolari; alcune di queste caratteristiche denotano periodicità, mentre altre comportano delle amplificazioni a determinate frequenze. Così, manipolando lo spettro di potenza delle primitive in input è possibile cambiare questi descrittori qualitativi. Diverse tecniche di filtraggio possono essere impiegate a tale scopo:

- I filtri lineari FIR sono filtri facili da progettare e implementare e offrono una certa flessibilità. Il modellamento di un rumore bianco può ad esempio essere fatto mediante un banco di filtri FIR passabanda (equalizzatore parametrico), nel quale i parametri impostabili sono la frequenza centrale, la larghezza di banda e il guadagno.
- I filtri IIR consentono un miglior controllo spettrale, pur essendo più complessi da progettare. Quando in input ad un filtro IIR viene dato un rumore, il sistema è anche noto come modello ARMA (autoregressive moving average).

L'implementazione del filtraggio è computazionalmente piuttosto costosa. Tale problema può essere ovviato utilizzando un pre-filtraggio, il quale permette di risparmiare tempo durante la simulazione consumando però più memoria.

# Capitolo 5

## Il Phantom<sup>TM</sup>

### 5.1 Le interfacce Phantom

Lo sviluppo del Personal Haptic Interface Mechanism (Phantom) ha avuto inizio al MIT Artificial Intelligence Laboratory. L’interazione da parte dell’utente avveniva inserendo un dito in un apposito ditale (sistema che poi è stato sostituito dall’uso di uno stilo): il Phantom legge la posizione e in base a questa esercita una forza corrispondente sul dito.

Lo sviluppo del Phantom è basato su tre importanti osservazioni:

- *Le caratteristiche aptiche più importanti sono la forza e il moto.* Le informazioni su come un oggetto si muove in risposta ad una forza applicata e le forze che nascono dal tentativo di muoverlo possono essere sufficienti per capire la geometria (forma, posizione), le proprietà (frizione, elasticità) degli oggetti e gli eventi in un ambiente. Le interazioni aptiche, al contrario degli altri tipi di interazioni, permettono uno scambio di dati bidirezionale.
- *Molte interazioni aptiche significative non coinvolgono movimenti di torsione.* Tale osservazione ha portato all’uso del ditale, e pertanto il dito dell’utente può essere modellato come un punto o una piccola sfera nell’ambiente virtuale (lo stesso avviene se si usa uno stilo). Tutto ciò semplifica notevolmente sia la programmazione che la progettazione.
- *Uno spazio di lavoro piccolo e centrato sul polso è sufficiente.* Molte interazioni aptiche avvengono all’interno dello spazio che può essere coperto dal movimento delle dita, mentre l’avambraccio compie movimenti limitati. Dai risultati di alcuni esperimenti si è deciso di costruire il Phantom in modo tale che un utente possa muovere liberamente il polso senza uscire dallo spazio di lavoro (come avviene per i mouse pad e le tastiere).



**Figura 5.1:** Il dispositivo aptico Phantom® Omni™ di SensAble Technologies.

Inoltre il design e la progettazione sono state pensate nel rispetto di tre regole fondamentali per garantire una riflessione delle forze corretta ed efficiente:

- *Il movimento libero deve essere avvertito come tale* (il dispositivo non deve esercitare forze esterne sull'utente).
- *Gli oggetti virtuali solidi devono essere avvertiti come oggetti rigidi.* Il Phantom (nel modello Omni™) è in grado di riflettere una rigidità massima di circa 35 N/cm, sufficienti per simulare una resistenza da parte degli oggetti rigidi al tocco.
- *I vincoli virtuali non possono essere violati.* Ad esempio non può accadere che, imprimendo una grande forza contro un muro virtuale, l'utente sia in grado di attraversarlo.

Il Phantom® Omni™ è una delle interfacce aptiche più economiche attualmente disponibili sul mercato. Si tratta di un dispositivo aptico ground-based a sei gradi di libertà in input, in grado di leggere la posizione sui tre assi principali  $x$ ,  $y$  e  $z$  e di computare le forze lungo gli stessi. È compatibile con tutti i PC intel-based ed è dotato di una connessione FireWire® IEEE-1394a che assicura un trasferimento dei dati ad alta velocità.

## 5.2 Il toolkit OpenHaptics™

La programmazione del Phantom® Omni™ avviene tramite il toolkit *OpenHaptics™* di SensAble. Tale toolkit include:

**HDAPI (Haptic Device API)** – Costituiscono lo strato di livello più basso per la programmazione aptica; permettono il rendering diretto delle forze e offrono un controllo sulla configurazione in runtime.

**HLAPI (Haptic Library API)** – Si appoggiano alle HDAPI per fornire un controllo di più alto livello; risultano familiari a chi già conosce l'OpenGL e sono più facili da usare in quanto il programmatore non deve preoccuparsi di eventi critici come il rendering di equazioni fisiche o la sicurezza dei thread.

Le HDAPI possono essere usate per richiedere le proprietà del dispositivo, come i gradi di libertà in input e output, la forza nominale massima, le dimensioni dello spazio di lavoro. Devono essere utilizzate per inizializzare e configurare l'HHD (haptic device handle) e inoltre possono essere usate per modificare la frequenza del *servo loop*; il servo loop è il ciclo di controllo usato per calcolare le forze da inviare al dispositivo aptico. Per avere un feedback stabile, il ciclo deve essere eseguito ad una frequenza pari a 1 KHz o superiore (maggiore è la frequenza, maggiore è l'utilizzo di CPU), e pertanto viene eseguito in un thread separato ad alta priorità (*servo thread*).

Le HLAPI permettono l'aggiunta di effetti personalizzati, ovvero l'aggiunta di forze da inviare al dispositivo aptico. Dato che le forze sono computate nel servo thread, possono essere usate in aggiunta alle callback HLAPI per avere ulteriori informazioni sul dispositivo.

## 5.3 Creazione di un ambiente aptico

Nei dispositivi aptici le forze sono usate per resistere o assistere il movimento del dispositivo stesso. Le interazioni delle forze derivano dal considerare la posizione del dispositivo in relazione agli oggetti presenti nell'ambiente virtuale: se la forza è nulla il movimento del dispositivo è libero. Quando l'utente muove il dispositivo, viene effettuato il rendering delle forze alla frequenza di 1000 Hz, impedendo all'*end effector* di penetrare la superficie degli oggetti; il modo in cui vengono renderizzate varia a seconda dell'effetto che si deve ottenere sulle superfici (dure, soffici, elastiche, ruvide) o sull'ambiente (come viscosità e inerzia). Un altro effetto desiderato potrebbe essere quello di costringere il movimento del dispositivo lungo un determinato percorso.

Il vettore delle forze è l'unità di output per un dispositivo aptico; le tre classi principali di forze sono:

- Forze dipendenti dal moto.
- Forze dipendenti dal tempo.
- Forze dipendenti sia dalla posizione che dal tempo.

### 5.3.1 Forze dipendenti dal moto

Una forza è dipendente dal moto quando viene calcolata in base al movimento del dispositivo aptico; tali forze sono:

**Forza elastica** – La forza elastica può essere calcolata usando la legge di Hooke

$$F = kx,$$

dove  $k$  è la costante elastica e  $x$  è il vettore della posizione. La molla che rappresenta la forza è attaccata, ad una estremità, ad un punto fisso  $p_0$ , solitamente collocato sulla superficie dell'oggetto che l'utente sta toccando, mentre l'altra estremità coincide con la posizione  $p_1$  del dispositivo; il vettore  $x = p_0 - p_1$  è orientato in modo tale che la forza elastica sia sempre diretta verso il punto fisso.

**Frizione viscosa** – L'utilità principale della frizione viscosa è ridurre la vibrazione opponendosi al moto; in generale tale forza è proporzionale alla velocità del dispositivo:

$$F = -bv,$$

dove  $b$  è la costante di smorzamento e  $v$  la velocità del dispositivo.

**Frizione statica** – E' una forza che si oppone alla direzione del moto con modulo costante:

$$F = -c \cdot sgn(v),$$

dove  $v$  è la velocità del dispositivo e  $c$  è la costante di frizione, dipendente dalla forza normale. Può essere utile per creare una transizione smorzata nei cambiamenti di direzione.

**Frizione dinamica** – E' simile alla frizione viscosa, e pertanto viene calcolata mediante la formula

$$F = -bv,$$

dove  $b$  dipende dalla forza normale.

**Inerzia** – E’ associata alla massa in movimento e si calcola, una volta nota la traiettoria, con la legge di Newton  $F = ma$ .

### 5.3.2 Forze dipendenti dal tempo

Le forze funzioni del tempo possono essere suddivise in:

**Costanti** – Si tratta di forze costanti sia in modulo che in direzione. Possono essere usate per far sentire il dispositivo più pesante oppure, al contrario, più leggero effettuando una compensazione della forza di gravità.

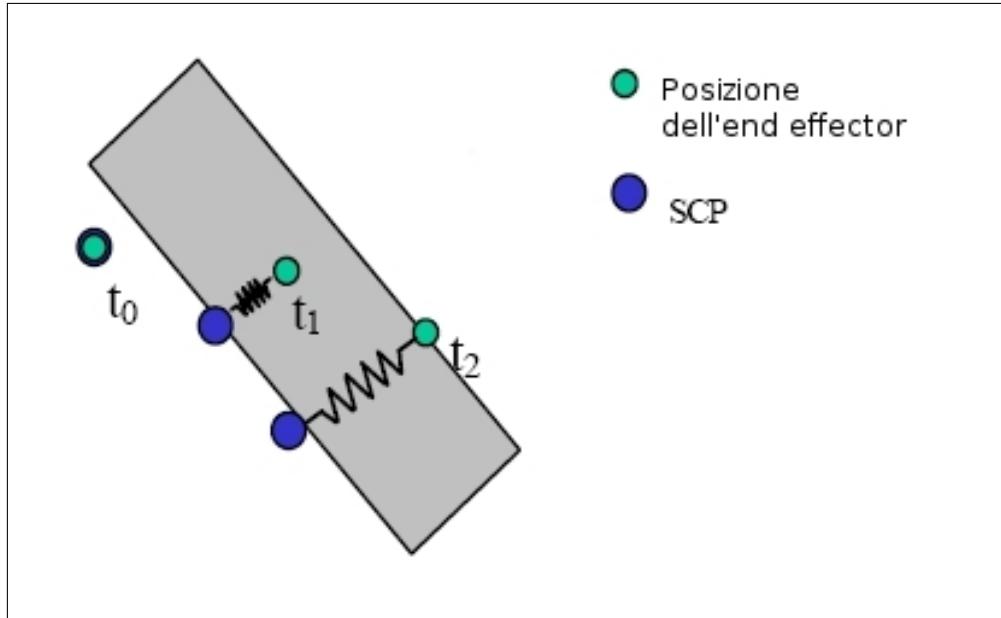
**Periodiche** – Derivano dall’applicazione di un pattern (dente di sega, onda quadra, sinusoide) che si ripete nel tempo. Una forza periodica è descritta da una direzione, da una costante di tempo che controlla il periodo del pattern e un’ampiezza che determina quanto forte dovrà essere la forza al suo picco massimo.

**Impulsive** – Sono costituite da un vettore delle forze che viene applicato istantaneamente; nella pratica, un impulso con un dispositivo aptico è applicato su un piccolo intervallo di tempo.

## 5.4 Contatti e vincoli

Simulare contatti con un oggetto virtuale significa computare le forze che impediscono all’*end-effector* del dispositivo di penetrare la superficie dell’oggetto virtuale. Ciò può essere simulato tramite il concetto di *proxy* che segue la trasformazione dell’end–effector nell’ambiente virtuale.

Il proxy è identificato con un punto, una sfera o un insieme di punti; se si tratta di un punto lo definiamo *SCP* (*surface contact point* o punto di contatto di superficie, figura 5.2). Quando l’end–effector penetra la superficie, viene calcolata una trasformazione del proxy che raggiunga la configurazione ad energia minima tra la superficie di contatto e l’end–effector. Successivamente vengono determinate le forze che impediscono al moto del dispositivo di penetrare ulteriormente la superficie, usando un controllo di elasticità–smorzamento; più precisamente, il punto SCP è un punto che segue la posizione dell’end–effector pur essendo costretto a restare sulla superficie dell’oggetto. Nello spazio libero l’SCP si trova nella stessa posizione dell’end–effector ( $t_0$  in figura 5.2); al momento del contatto con un oggetto l’SCP può essere calcolato muovendo l’ultimo SCP verso la posizione dell’end–effector senza oltrepassare la superficie. La forza viene calcolata simulando una molla collegata da un lato all’SCP e dall’altro alla posizione dell’end–effector: in figura 5.2  $t_1$  indica la penetrazione nell’oggetto e  $t_2$  mostra



**Figura 5.2:** Rappresentazione del punto SCP (surface contact point).

una penetrazione ulteriore (la molla è maggiormente allungata facendo sentire una maggiore resistenza all’utente).

## 5.5 Sincronizzazione

La sincronizzazione è importante quando l’interfaccia utente è composta sia di una parte grafica che di una parte aptica, in quanto i due cicli di rendering devono accedere alle stesse informazioni; ciò implica la creazione di copie dei dati in memoria, resi così disponibili in modo sicuro ad entrambi i thread. Non può essere usata la mutua esclusione in primo luogo perché il ciclo di rendering aptico deve sempre girare ad una frequenza di 1000 Hz e non può quindi attendere altri processi, in secondo luogo perché la diversa frequenza dei due cicli facilita la presenza di inconsistenze se sono in movimento più oggetti contemporaneamente.

Anche nella gestione degli eventi il ciclo di rendering aptico deve avere la maggiore priorità. Quando si verifica un evento (come il tocco di una superficie o l’applicazione di un particolare vincolo), questo viene prima gestito dal thread aptico, in modo da fornire una risposta al dispositivo immediata, e poi accodato dal thread grafico che si occuperà di aggiornare la visualizzazione sullo schermo al successivo frame.

## 5.6 Convenzioni nell'interfaccia grafica e aptica

**Pozzo di gravità** – Il pozzo di gravità viene usato per attirare il dispositivo verso un determinato punto, solitamente indicato come *vincolo istantaneo*. Al pozzo è associato un raggio di influenza: quando il dispositivo si trova all'interno di tale raggio, viene applicata una forza che lo attira verso il centro del pozzo (solitamente viene usata una forza elastica).

**Trasformazioni relative** – I dispositivi aptici hanno una posizione assoluta, dal momento che si trovano ancorati al tavolo di lavoro. L'unico modo per ottenere una manipolazione relativa è applicare delle trasformazioni addizionali alle coordinate del dispositivo, così da dare l'impressione che quest'ultimo si stia muovendo relativamente ad una data posizione e orientazione.

**Accoppiamento delle informazione aptiche e visive** – Il senso aptico del contatto può essere migliorato fornendo una rappresentazione visuale del contatto stesso; ciò si ottiene semplicemente dando una corretta visuale. Ad esempio, la sensazione di contatto sarà più verosimile se il cursore non penetra mai la superficie (rappresentando quindi il proxy e non la posizione del dispositivo).

**Stabilizzazione della manipolazione con la frizione** – Applicare una piccola frizione aiuta l'utente a stabilizzare la mano mentre cerca di arrivare alla posizione desiderata; senza frizione, il dispositivo è troppo “libero”, rendendo difficile effettuare posizionamenti precisi.

## 5.7 Programmare con le HD API

Le Haptic Device API consistono di due componenti: le API relative al dispositivo e quelle relative allo scheduler. Le prime si occupano dell'astrazione di ogni meccanismo tridimensionale supportato. Le API dello scheduler invece permettono di introdurre dei comandi che verranno eseguiti all'interno del servo loop thread. Il tipico uso delle HD API prevede l'inizializzazione del dispositivo e dello scheduler, l'avvio di quest'ultimo, l'esecuzione di alcuni comandi tramite lo scheduler stesso e l'uscita. Ad esempio, consideriamo la creazione di un piano che respinga il dispositivo quando questo cerca di penetrarlo; la creazione di tale ambiente si divide in 5 passi:

1. Inizializzazione del dispositivo.

2. Creazione delle callback dello scheduler che chiedono la posizione del dispositivo e comandano la forza che deve respingere il dispositivo al momento della penetrazione.
3. Abilitazione delle forze del dispositivo.
4. Avvio dello scheduler.
5. Reset del dispositivo e dello scheduler quando l'applicazione viene terminata.

Le routine relative al dispositivo possono essere raggruppate come segue:

- Inizializzazione del dispositivo (creazione dell'handle del dispositivo, abilitazione delle forze, calibrazione).
- Sicurezza del dispositivo (controllo della sicurezza del force feedback, come temperatura dei motori, forze e velocità eccessive).
- Stato del dispositivo (richiesta di posizione, velocità, pulsanti e matrici di trasformazione).

Un tipico schema di programmazione con le HD API è rappresentato in figura 5.3.

### 5.7.1 Operazioni del dispositivo

Le operazioni di dispositivo sono tutte quelle inerenti la richiesta e l'impostazione dello stato corrente; sono tutte operazioni che dovrebbero essere eseguite esclusivamente all'interno del servo loop utilizzando le callback dello scheduler.

**Inizializzazione** – Sia il dispositivo che lo scheduler devono essere inizializzati prima dell'uso; al momento dell'inizializzazione le forze sono disattivate, verranno invece attivate nel momento in cui verrà fatto partire lo scheduler. Se vengono utilizzati più dispositivi, ognuno deve essere inizializzato separatamente, mentre viene avviato un solo scheduler.

**Dispositivo corrente** – Nel caso in cui vengano utilizzati più dispositivi, uno deve essere impostato come il dispositivo corrente; per fare ciò viene usato il comando seguente:

```
mdMakeCurrentDevice(hHD)
```

Nell'utilizzo di un singolo dispositivo tale comando non deve mai essere utilizzato.

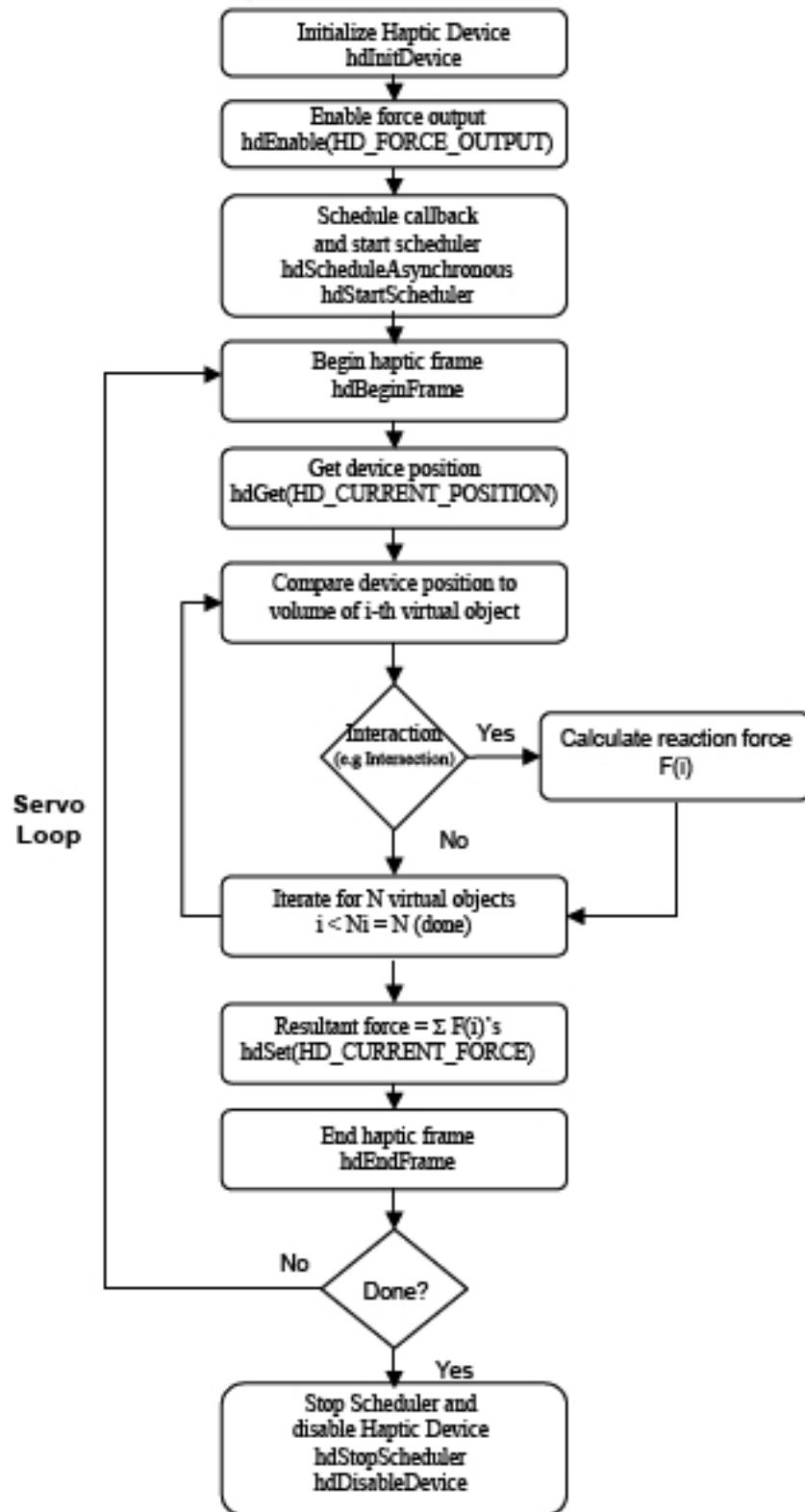


Figura 5.3: Schema di programmazione con le HD API.

**Caratteristiche del dispositivo** – Alcune caratteristiche del dispositivo possono essere abilitate o disabilitate a seconda delle necessità, usando i comandi `hdEnable` e `hdDisable`. Tali istruzioni devono essere utilizzate con attenzione e sempre tramite callback dello scheduler.

### 5.7.2 Frame aptici

I frame aptici definiscono i limiti all'interno dei quali lo stato del dispositivo è consistente. All'avvio del frame, lo stato del dispositivo viene aggiornato e memorizzato per l'uso in quel frame; successivamente tutte le operazioni dovrebbero essere eseguite all'interno dello stesso frame, dato che cercando di ottenere lo stato del dispositivo al di fuori del frame si può ottenere invece lo stato del frame precedente. Ad ogni istante lo scheduler dovrebbe avere un solo frame attivo per ogni dispositivo; tuttavia i frame per dispositivi diversi possono essere annidati.

### 5.7.3 Operazioni dello scheduler

Come già detto, lo scheduler si occupa della gestione del servo loop thread e opera ad una frequenza di circa 1000 Hz; diventa perciò pericoloso accedere manualmente alle variabili: per fare ciò conviene sempre utilizzare le callback. Le chiamate allo scheduler sono di due tipi:

**Chiamate sincrone** – Ritornano solamente quando sono complete, e il thread deve attendere il completamento prima di continuare. Sono usate principalmente per richiedere lo stato del sistema.

**Chiamate asincrone** – Ritornano non appena sono state schedulate. Sono usate per rappresentare un effetto aptico e perciò risiedono nello scheduler, applicando l'effetto ad ogni iterazione. Al momento della schedulazione di una callback asincrona viene restituito un handle che può essere riutilizzato successivamente per eseguire operazioni sulla callback, come l'eliminazione della stessa dallo scheduler o il suo blocco fino al completamento.

Tutte le callback hanno associata una priorità, in base alla quale viene determinato l'ordine di esecuzione all'interno dello scheduler; per ogni ciclo di scheduler viene eseguita sempre ogni callback. In ogni momento viene eseguito un solo thread di schedulazione e, se sono presenti più dispositivi, questi condividono lo stesso thread.

### 5.7.4 Stato del sistema

#### Ottenimento dello stato

Lo stato del dispositivo (assieme ad altre informazioni) può essere richiesto tramite l'uso delle funzioni della famiglia `hdGet`, come ad esempio `hdGetDoublev`. Tali funzioni richiedono un parametro valido ed un singolo indirizzo di ritorno o un array. Le richieste possono essere fatte al frame corrente o al precedente; in generale, se vengono eseguite all'esterno di un frame, vengono riferite a quello precedente. Per i parametri di forze in output il valore viene impostato a zero automaticamente all'inizio di ogni frame.

#### Impostazione dello stato

L'impostazione dello stato deve essere sempre eseguita all'interno di uno stesso frame, ed è necessario passare il numero corretto di parametri; questo per evitare l'introduzione di errori dato che si va a modificare le caratteristiche o il comportamento del dispositivo. Le forze non vengono inviate al dispositivo fino alla fine del frame, quindi se viene impostato due volte lo stesso stato, la seconda impostazione sostituisce la prima (se ad esempio si vogliono sommare più forze, sarà necessario farlo in una variabile separata).

#### Sincronizzazione dello stato

Lo scheduler fornisce una sincronizzazione dello stato tra thread diversi. Un esempio è costituito dal caso in cui uno stato deve essere aggiornato alla frequenza del servo loop, e contemporaneamente un altro thread (come il thread grafico) accede e modifica lo stato. E' possibile ottenere lo stato attuale da un singolo frame all'interno del ciclo aptico usando le callback sincrone.

## 5.8 Programmare con le HLAPI

Le HLAPI sono delle API in C di alto livello per il rendering aptico e si accompagnano alle API OpenGL per il rendering grafico. Le HLAPI permettono al programmatore di specificare primitive geometriche come triangoli, linee e punti assieme a proprietà aptiche come frizione e rigidità; tramite tali informazioni il motore di rendering aptico calcola poi le forze appropriate. Inoltre queste API permettono sia di impostare che di richiedere lo stato degli oggetti, richiedere lo stato del Phantom (posizione e orientamento) e impostare le funzioni di callback.

### 5.8.1 Generazione delle forze

Esistono tre modi per generare un feedback aptico usando le HLAPI:

**Rendering delle forme** – Permette di specificare primitive geometriche (tramite istruzioni OpenGL) che il motore di rendering usa per computare le giuste forze di reazione per simulare il tocco della superficie. L'identificazione delle geometrie create in OpenGL può avvenire in due modi: tramite il *depth buffer* oppure tramite il *feedback buffer*.

**Rendering degli effetti** – Permettono di specificare forze globali non definibili tramite primitive geometriche (cioè non legate al tocco di una figura geometrica).

**Rendering diretto del proxy** – Permettono di impostare un orientamento per il dispositivo aptico il quale verrà portato nella giusta posizione dal motore di rendering.

### 5.8.2 Threading

Dato che il rendering aptico necessita di un aggiornamento molto più frequente rispetto all'applicazione grafica, il motore HLAPI crea due thread in aggiunta a quello dell'applicazione principale: il *servo thread* e il *collision thread*.

**Servo thread** – Gestisce la comunicazione diretta con il dispositivo aptico, leggendo la posizione e l'orientamento del dispositivo e aggiornando le forze ad una frequenza molto alta (generalmente 1000 Hz). La differenza rispetto alle HD API è che le HL API nascondono il servo thread all'utente.

**Collision thread** – Determina quali primitive geometriche sono in contatto con il proxy ad una frequenza di 100 Hz (minore rispetto al servo thread ma maggiore del client thread). Una volta determinato quale oggetto è in contatto con il proxy, viene elaborata un'approssimazione della forma locale dell'oggetto, la quale viene inviata al servo thread che la utilizza nel calcolo delle forze.

### 5.8.3 Struttura della programmazione con le HLAPI

In figura 5.4 si può vedere la struttura tipica di un programma che implementa le HL API. Il primo passo è l'inizializzazione dell'ambiente OpenGL con la creazione di un contesto grafico e la rispettiva finestra; segue l'inizializzazione delle HL API con la creazione del contesto aptico. Successivamente viene specificato come le coordinate fisiche (quelle del Phantom) devono essere mappate nello

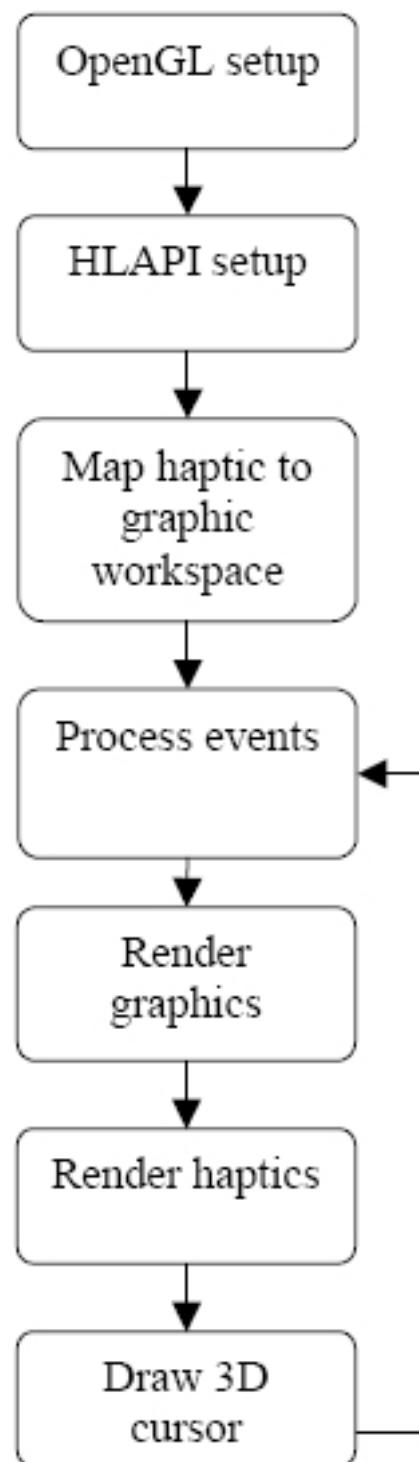


Figura 5.4: Schema di programmazione con le HLAPI.

spazio delle coordinate usato dall'ambiente grafico. A questo punto è possibile effettuare il rendering grafico e inizia la cattura degli eventi aptici, procedendo così con il rendering aptico; in aggiunta viene rappresentato un cursore tridimensionale in corrispondenza della posizione del proxy. Il ciclo prosegue tornando alla cattura degli eventi.

### 5.8.4 Setup del dispositivo

Come con le HDAPI, l'inizializzazione del dispositivo avviene tramite il comando `hdInitDevice`, seguito dal comando `hlCreateContext`, il quale crea il contesto aptico. Infine il contesto viene impostato come corrente con l'istruzione `hlMakeCurrent` (ciò è richiesto da tutti i comandi delle HLAPI). Il rendering del contesto deve essere attivo per un solo thread alla volta, in quanto le routine HLAPI, come quelle OpenGL, non sono *thread safe*; ciò può essere implementato con l'uso di un *mutex* per sincronizzare le chiamate a `hlMakeCurrent` per i contesti condivisi.

### 5.8.5 Frame aptici

Tutti i comandi implementati dalle HLAPI devono essere inseriti all'interno di un *frame aptico*, delimitato dalle chiamate a `hlBeginFrame` (posta all'inizio del ciclo di rendering) e `hlEndFrame` (alla fine del ciclo di rendering), come evidenziato in figura 5.5. In generale sarà presente un frame aptico per ogni frame grafico. Ciò è molto diverso rispetto alla programmazione con le HDAPI, perché in questo caso il frame grafico viene aggiornato alla stessa frequenza di quello aptico: le chiamate avvengono nel thread dato che entrambi accedono alle medesime informazioni geometriche. `hlBeginFrame` campiona lo stato corrente del rendering aptico dal thread aptico; successivamente `hlEndFrame` aggiorna la posizione del proxy in base ai cambiamenti nella dinamica degli oggetti. In aggiunta `hlBeginFrame` si occupa di aggiornare le coordinate globali usate dal motore di rendering aptico. Tutte le richieste dello stato del dispositivo o del proxy effettuate all'interno di uno stesso frame riportano lo stesso risultato, ovvero lo stato presente al momento in cui è stato richiamato `hlBeginFrame`. Alla fine del frame, tutti i cambiamenti allo stato che sono intervenuti vengono trasmessi al rendering aptico. Ciò fa sì che più cambiamenti allo scenario all'interno di uno stesso frame vengano trasmessi ai cicli aptico e grafico simultaneamente alla fine del frame.

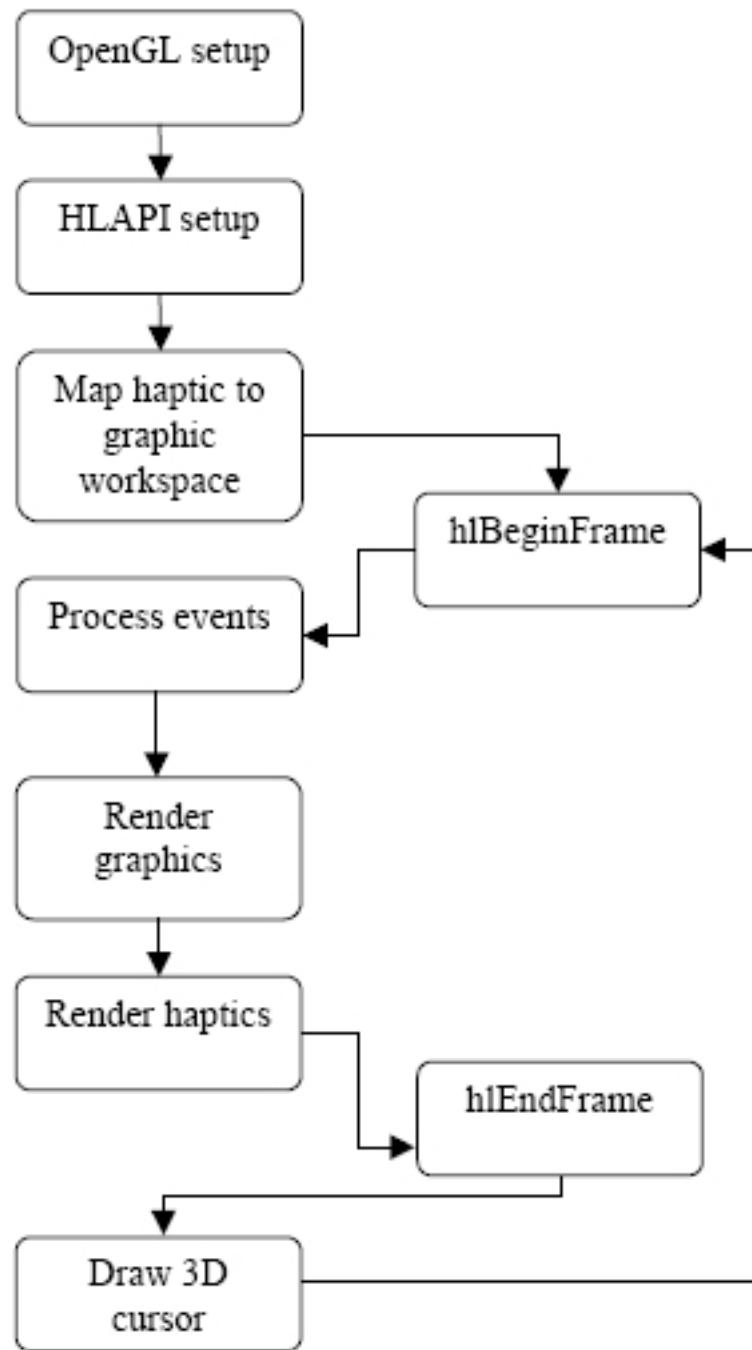


Figura 5.5: Collocazione dei frame con le HLAPI.

### 5.8.6 Rendering delle forme

Il rendering delle forme con le HLAPI è usato per rappresentare superfici e oggetti solidi. Una forma può essere creata combinando insieme più primitive grafiche come linee e poligoni.

#### Inizio e fine di una forma

Le forme geometriche vengono specificate attraverso istruzioni OpenGL racchiuse tra le chiamate a `hlBeginShape` e `hlEndShape`. Le HLAPI catturano tali geometrie per calcolare il rendering aptico.

#### Identifieri delle forme

Ogni forma deve essere identificata univocamente da un intero che sarà usato dal motore di rendering per i cambiamenti della forma da frame a frame in modo da calcolare correttamente le forze. Tale identificatore può essere rilasciato quando la forma non è più usata.

#### Depth buffer

La geometria degli oggetti può essere catturata dal *depth buffer* usato dalle OpenGL: quando viene richiamata l'istruzione `hlEndShape` le API leggono un'immagine da tale buffer, immagine che viene poi passata al collision thread e usata per calcolare le collisioni con il proxy. Ogni modifica al depth buffer verrà riconosciuta come modifica di un oggetto e quindi ne verrà fatto il rendering aptico. Dato che gli oggetti vengono convertiti in un'immagine, è fondamentale utilizzare il punto di vista corretto durante il rendering, in quanto non è possibile sentire quelle parti della scena che non sono visibili da tale punto. Esiste la possibilità di abilitare l'ottimizzazione del punto di vista, con la quale le HLAPI correggono automaticamente i parametri di visualizzazione OpenGL in base alla mappatura del dispositivo aptico sulla scena; il comando corrispondente è

```
hlEnable(HL_HAPTIC_CAMERA_VIEW)
```

Grazie al *depth buffer* è possibile effettuare il rendering aptico e grafico delle forme in un unico ciclo, semplicemente includendo il codice di rendering grafico all'interno di un blocco `hlBeginShape` - `hlEndShape`. Se è attiva l'ottimizzazione *haptic camera view*, tale metodo non funzionerà in quanto il rendering grafico verrà effettuato dal punto di vista modificato.

## Feedback buffer

Il rendering tramite *feedback buffer* usa il feedback buffer delle OpenGL per catturare le primitive geometriche: quando l'istruzione `hlBeginShape` viene richiamata, le HLAPI automaticamente allocano tale buffer e impostano la modalità di rendering OpenGL a feedback buffer; tutte le primitive vengono così salvate, ma solo i comandi che generano punti, linee o vertici vengono catturati dalla routine aptica (gli altri comandi, come quelli relativi all'impostazione delle texture, vengono ignorati). Con il comando `hlEndShape` le primitive salvate nel feedback buffer vengono usate per il rendering aptico. E' opportuno utilizzare l'istruzione `hlHinti` per impostare il numero di vertici che saranno presenti nella scena, numero che verrà usato dalle HLAPI per allocare la memoria per il feedback buffer; ad esempio il comando seguente alloca memoria per 4 vertici:

```
hlHinti(HL_SHAPE_FEEDBACK_BUFFER_VERTICES, 4);
```

il valore di default è 65536.

Durante la creazione di un oggetto con il feedback buffer è importante non richiamare l'istruzione `glCullFace` (o la relativa abilitazione/disabilitazione) all'interno della coppia di istruzioni `hlBeginShape` - `hlEndShape`, altrimenti può accadere di non essere in grado di percepire alcune parti dell'oggetto.

## Ottimizzazione del rendering

Come nella grafica è possibile ottimizzare il rendering effettuandolo solo sulle forme che vengono visualizzate al momento, è possibile ottimizzare il rendering aptico effettuandolo solo sulle forme che possono essere toccate (ad esempio considerando solo le parti degli oggetti vicini alla posizione corrente del proxy). Di seguito è riportato un elenco delle possibili ottimizzazioni che possono essere effettuate.

**Adaptive Viewport** – Tale ottimizzazione consiste nel limitare la regione del depth buffer che viene letta nella memoria per il rendering aptico all'area prossima alla posizione corrente del proxy. L'incremento di prestazioni dipende dalla velocità con la quale la scheda grafica riesce a leggere il depth buffer dalla memoria. Il comando per abilitare tale opzione è

```
hlEnable(HL_ADAPTIVE_VIEWPORT)
```

Per utilizzare l'adaptive viewport la scena deve essere ridisegnata regolarmente quando il dispositivo aptico è in movimento, ad una frequenza tanto più alta quanto più veloce è il movimento del dispositivo.

**Haptic Camera View** – Con tale opzione attivata, le HLAPI modificano automaticamente i parametri di visualizzazione usati per il rendering del depth buffer o del feedback buffer, in modo tale che solo le forme vicino alla posizione attuale del proxy vengono visualizzate. Nel caso venga utilizzato il feedback buffer, tale tecnica può portare ad un incremento delle prestazioni in quanto viene ridotto il numero delle primitive geometriche prese in considerazione dal rendering aptico (l'incremento effettivo dipende dalla densità degli oggetti presenti sulla scena). Nel caso invece di utilizzo del depth buffer l'incremento sarà di minore entità, in quanto il rendering aptico del depth buffer è indipendente dal numero di primitive; inoltre l'immagine generata dal depth buffer è solo un sottoinsieme dell'intero buffer ed è possibile sentire parti degli oggetti che non sono visibili dal punto di vista grafico. Come per l'adaptive viewport, l'abilitazione dell'opzione *haptic camera view* richiede che la scena venga ridisegnata ad una frequenza proporzionale alla velocità di movimento del dispositivo.

L'utilizzo di questa opzione disabilita l'adaptive viewport.

**Culling con partizioni spaziali** – Quando nella scena sono presenti degli oggetti costituiti da un numero molto grande di primitive il culling può diventare molto costoso in termini di risorse di calcolo; usando strutture particolari (come alberi binari) è possibile effettuare il culling di molte primitive in un'unica operazione.

Per prima cosa è necessario determinare la regione dello spazio che viene presa in considerazione per il rendering aptico; tale regione è semplicemente quella che viene impostata dalle HLAPI nella chiamata a `hlBeginShape` quando è abilitata la *haptic camera view*. Una volta individuata la regione, si utilizzano delle partizioni dello spazio per trovare il sottoinsieme di primitive che si trovano all'interno (completamente o parzialmente) di essa; tali geometrie vengono visualizzate tramite OpenGL.

In conclusione, se si lavora con un numero elevato di primitive è conveniente utilizzare il depth buffer mentre, al contrario, è conveniente utilizzare il feedback buffer se il numero di primitive è ridotto.

Il depth buffer è meno accurato del feedback buffer, anche se tale differenza non è percettibile: con il depth buffer infatti le forme vengono trasformate in un'immagine bidimensionale prima di calcolarne il rendering aptico, e tale trasformazione comporta una perdita di informazioni. L'opzione *haptic camera view* permette di individuare una vista che minimizzi la perdita di dettagli dell'immagine, anche se non sempre è possibile catturare tutte le informazioni. Se vengono



**Figura 5.6:** Mappatura dallo spazio di lavoro aptico alla scena grafica.

usate linee o punti come vincoli, deve essere usato il feedback buffer in quanto tali primitive non possono essere catturate dal depth buffer.

### 5.8.7 Mappatura del dispositivo aptico sulla scena

Di seguito viene descritto come i movimenti del dispositivo aptico vengono tradotti in movimenti nella rappresentazione grafica.

#### Lo spazio di lavoro aptico

Lo spazio di lavoro aptico è lo spazio che può essere raggiunto dal dispositivo aptico, e le dimensioni (in millimetri) di tale spazio possono essere ottenute con il comando

```
HLdouble workspaceDims[6];
hlGetDoublev(HL_WORKSPACE, workspaceDims);
```

Si può scegliere di non utilizzare l'intero spazio di lavoro scegliendo la porzione utilizzabile con la chiamata a `hlWorkspace`.

#### Stack di matrici

Sono previsti due stack di matrici 4x4 tramite le quali viene effettuata la mappatura: `HL_VIEWTOUCH_MATRIX` e `HL_TOUCHWORKSPACE_MATRIX`. Mentre il primo ha la funzione di definire una mappatura tra lo spazio di lavoro e le coordinate visive, il secondo serve ad orientare lo spazio di lavoro secondo le coordinate visive (figura 5.6).

Le coordinate globali vengono prima trasformate nelle coordinate locali (le coordinate della vista corrente); successivamente si ottengono le coordinate di tocco, le quali rappresentano la base della mappatura dello spazio di lavoro sulle coordinate della vista corrente. Infine l'ultima trasformazione porta ad ottenere le coordinate dello spazio di lavoro (coordinate locali del dispositivo aptico).

Il funzionamento dei due stack è analogo a quello delle OpenGL: le HLAPI mantengono uno stack corrente e tutte le operazioni hanno effetto su questo.

### 5.8.8 Proprietà dei materiali ed effetti aptici

Controllando le proprietà dei materiali si controllano le proprietà tattili della superficie, analogamente a come vengono controllate le proprietà visive. Il comando per specificare tali proprietà è `hlMaterial`, e può essere applicato al fronte o retro della superficie o ad entrambe le facce.

**Rigidità** – La rigidità di una superficie viene impostata tramite il comando:

```
hlMaterialf(HL_FRONT_AND_BACK, HL_STIFFNESS, 0.7);
```

dove la `f` aggiunta indica che il parametro numerico è un *float*. Matematicamente rappresenta il tasso con il quale la forza aumenta mano a mano che il dispositivo tenta di penetrare la superficie secondo la legge di Hooke  $F = kx$ , dove  $k$  è la costante di rigidità e  $x$  è il vettore rappresentante la penetrazione.

**Smorzamento** – Impostando tale proprietà tramite il comando:

```
hlMaterialf(HL_FRONT_AND_BACK, HL_DAMPING, 0.1);
```

viene aggiunta una forza di resistenza dipendente dalla velocità secondo la legge  $F = kv$ , con  $k$  costante di smorzamento e  $v$  velocità del dispositivo.

**Frizione** – La frizione indica la resistenza di un oggetto al movimento laterale su di esso. Si distingue tra frizione *statica* e *dinamica*: la prima rappresenta la resistenza che si avverte quando il dispositivo inizia il suo moto sull'oggetto, mentre la seconda indica la resistenza che si avverte quando il dispositivo è in movimento sull'oggetto. Si può pensare al ghiaccio come esempio di materiale dotato di un'alta frizione statica ma bassa frizione dinamica; la gomma invece ha coefficienti elevati per entrambi i tipi di frizione. Il comando per settare le due proprietà è:

```
hlMaterialf(HL_FRONT_AND_BACK, HL_STATIC_FRICTION, 0.3);
hlMaterialf(HL_FRONT_AND_BACK, HL_DYNAMIC_FRICTION, 0.2).
```

**Vincoli** – Sulla superficie possono essere specificati dei vincoli che costringono il proxy in una determinata posizione (come nella simulazione di una superficie magnetica). Per attivare questa modalità è sufficiente eseguire il seguente comando prima della creazione dell'oggetto:

```
hlTouchModel(HL_CONSTRAINT);
```

per tornare alla modalità di default si richiama nuovamente la funzione `hlTouchModel` con il parametro `HL_CONTACT`. E' possibile inoltre impostare il raggio d'azione del vincolo tramite il comando:

```
hlTouchModelf(HL_SNAP_DISTANCE, 1.5);
```

oggi volta che il proxy viene a trovarsi ad una distanza dal vincolo minore del valore specificato da questa istruzione, verrà applicato il vincolo con una forza proporzionale alla distanza.

Per tutte le proprietà il coefficiente varia da 0 a 1. E' opportuno porre attenzione all'utilizzo di valori troppo elevati in quanto possono causare instabilità del dispositivo.

### 5.8.9 Eventi

Tramite le HLAPI è possibile associare una funzione al verificarsi di un determinato evento. Ciò avviene eseguendo il comando:

```
hlAddEventCallback(HL_EVENT_TOUCH,  
    HL_OBJECT_ANY, HL_CLIENT_THREAD, &function, NULL);
```

in questo modo verrà eseguita la funzione `function` ogni volta che un'oggetto verrà toccato. La funzione di callback deve essere specificata come segue:

```
void HLCALLBACK function(HLENUM event,  
    HLuint object, HLEnum thread, HLcache *cache,  
    void *userdata)
```

Le callback possono essere associate ai seguenti eventi:

**Tocco** – Un tocco (*touch*) viene individuato quando il motore di rendering determina un contatto tra il proxy e un oggetto di cui è stato fatto il rendering nell'ultimo frame; se il proxy resta in contatto con l'oggetto, solo il primo istante di contatto viene rilevato come tocco. Analogamente il distacco (*untouch*) dalla superficie viene rilevato quando il proxy smette di essere in contatto con l'oggetto.

**Movimento** – Il moto viene rilevato come tale quando la posizione o l'orientamento del proxy cambiano. La variazione di movimento e orientamento che viene rilevata come movimento può essere modificata tramite le variabili `HL_EVENT_MOTION_TOLERANCE` e `HL_EVENT_ANGULAR_TOLERANCE`.

**Pulsante** – Possono essere associate delle funzioni anche alla pressione dei pulsanti presenti sul dispositivo usando come identificatori per il primo pulsante `HL_EVENT1_BUTTON_DOWN`, `HL_EVENT1_BUTTON_UP` e per il secondo pulsante `HL_EVENT2_BUTTON_DOWN`, `HL_EVENT2_BUTTON_UP`.

Nel prossimo capitolo si vedrà come le HD API e le HL API sono state utilizzate per la creazione di un'applicazione diretta alla simulazione aptica di superfici ruvide.

# Capitolo 6

## L'applicazione Phantom Friction

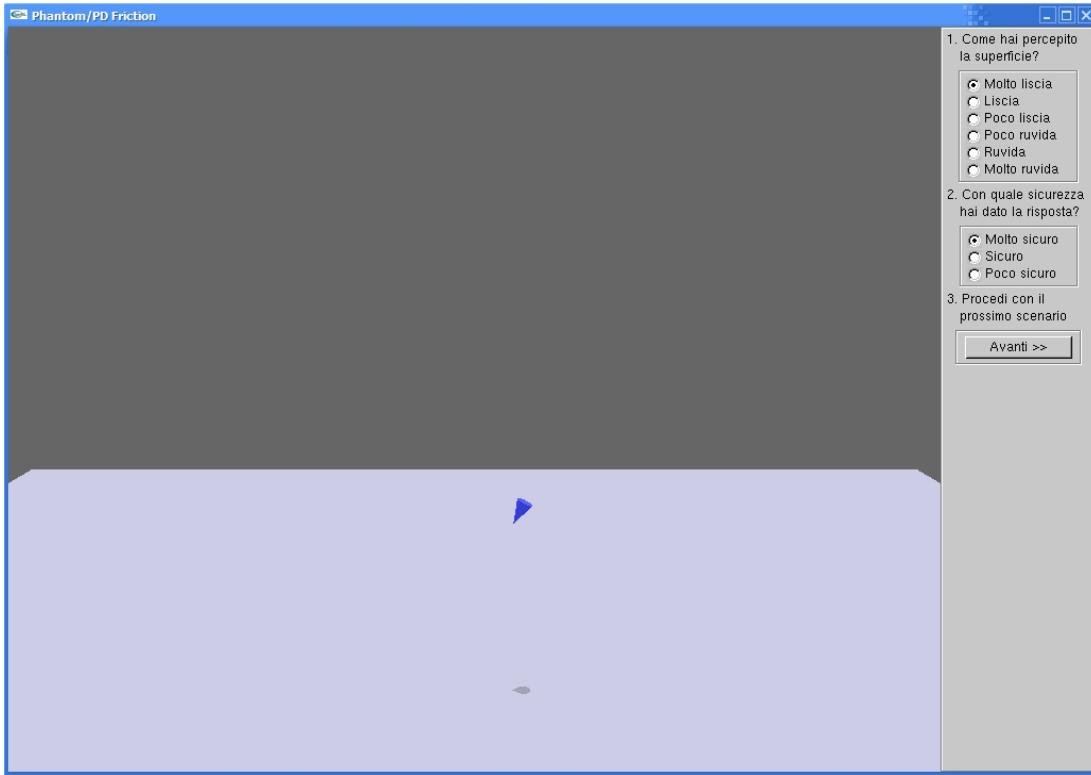
### 6.1 Scopo del lavoro

Per poter integrare la simulazione aptica di una tessitura con la simulazione sonora si è reso necessario scrivere un'applicazione che implementi tutti i concetti visti finora.

Dato che nei nostri scopi non rientra la realizzazione di particolari tessiture grafiche o nessun altro tipo di algoritmo di computer graphics, l'interfaccia (mostrata in figura 6.1) risulta molto semplice; tuttavia è chiara, user-friendly ed efficace (come vedremo) per un'interazione da parte dell'utente di tipo bimodale. Il nostro obiettivo infatti è permettere all'utente di *sentire* uno o più oggetti virtuali utilizzando i sensi del tatto e dell'udito; non vogliamo assolutamente fornire suggerimenti di tipo grafico, perché, come molti studi di Lederman e Klatzky [20] hanno dimostrato, ciò falserebbe la percezione aptica e uditiva. Secondo tali ricerche il senso della vista quasi sempre prevale sugli altri sensi quando è utilizzato per riconoscere oggetti. In particolare sembra che la vista prevalga soprattutto quando è richiesto di riconoscere le proprietà macroscopiche di un oggetto, come la sua forma o le tessiture che caratterizzano la sua superficie; il tatto invece viene usato quasi quanto la vista nel riconoscimento della ruvidità.

L'applicazione Phantom Friction è stata realizzata in C/C++, su sistema operativo Microsoft Windows. Le librerie utilizzate sono tutte multipiattaforma, quindi con poche modifiche al codice è possibile effettuare il porting su altri sistemi operativi, come Linux. Allo stesso modo le patch per Pure Data possono essere riutilizzate in Linux.

Tale applicazione implementa solamente la simulazione grafica e aptica, non la simulazione audio. Per quanto riguarda quest'ultima si sono sfruttate le patch in Pure Data già descritte: una loro nuova implementazione in un linguaggio di programmazione quale il C++ sarebbe stata problematica, in quanto sarebbe



**Figura 6.1:** Interfaccia grafica dell'applicazione Phantom Friction.

stato necessario riscrivere non solo gli algoritmi, ma anche tutte le funzioni già implementate come primitive in PD. D'altro canto non è stato possibile implementare le parti grafica e aptica in PD. Come si è detto, esistono librerie come GEM che permettono di unire la simulazione grafica a quella audio con l'utilizzo di semplici moduli che realizzano oggetti tridimensionali in OpenGL, ma non è possibile l'integrazione con le funzioni aptiche. Si potrebbe anche pensare di realizzare un oggetto per PD, appoggiandosi eventualmente alla libreria flex, che integri le funzioni aptiche e grafiche, ma ciò introducerebbe un overhead troppo elevato, tanto da rendere la simulazione troppo pesante in termini di risorse di calcolo anche per computer potenti. Si vede quindi che la soluzione di realizzare da un lato la simulazione grafica/aptica in C/C++ e dall'altro la simulazione audio in PD risulta la scelta migliore.

## 6.2 Implementazione grafica

La simulazione grafica è stata realizzata in OpenGL utilizzando le librerie incluse nel toolkit OpenHaptics™. Dal momento che queste librerie sono state precompilate per essere usate con l'ambiente di programmazione *Visual Studio .NET* di Microsoft, è necessario ricompilarle se devono essere usate con un altro ambiente. Nel nostro caso l'applicazione è stata sviluppata in *Visual Studio 2005*:

perciò è stato necessario effettuare il download del pacchetto `glui_v2_2.zip` (<http://glui.sourceforge.net/>); una volta scompattato il file, sarà presente una directory `msvc` contenente i file dei progetti da aprire con Visual Studio e ricompilare.

### 6.2.1 Rappresentazione della scena

Tramite un parallelepipedo è stata rappresentata la superficie sulla quale un cursore a forma di piccolo cono (controllato tramite il Phantom® Omni™) può muoversi; questa superficie è piana, di un solo colore, priva di texture e non viene mai variata, in accordo con la condizione secondo la quale non devono essere dati suggerimenti visivi sul tipo di superficie. Il parallelepipedo viene creato come lista composta da un solo oggetto:

```
DraggableObject dro;
dro.displayList = glGenLists(1);
dro.transform = hduMatrix::createTranslation(surfx,surfy,surfz);
glNewList(dro.displayList, GL_COMPILE);
drawSurface(1000.0f, 800.0f, 20.0f, texturized);
glEndList();
draggableObjects.push_back(dro);
```

dove il metodo `drawSurface` costruisce il parallelepipedo specificandone i vertici, accettando come argomenti le dimensioni del parallelepipedo lungo gli assi *x*, *z* e *y* e un valore booleano che indica se all'oggetto deve essere applicata la texture grafica associata (tale valore è sempre impostato a falso nell'attuale implementazione). Il comando

```
glCallList(obj.displayList);
```

effettua il rendering a video della superficie. E' possibile aggiungere alla lista altri oggetti, nel caso ad esempio si volessero rappresentare più superfici o aggiungere poligoni in modo da rendere più complessa la forma della superficie; le modifiche alle proprietà aptiche e grafiche effettuate sulla lista verranno propagate contemporaneamente a tutti gli oggetti della lista. Anche il cursore, rappresentato da un cono, viene disegnato come una lista costituita da un oggetto singolo tramite la funzione:

```
redrawCursor(const boolean& h, const int& nShadow);
```

il parametro booleano `h` deve essere impostato a 1 se si vuole che venga fatto sia il rendering grafico che aptico del cursore, mentre impostandolo a 0 verrà fatto

solo il rendering grafico; il secondo parametro è utilizzato nel calcolo delle ombre. Le istruzioni per il rendering grafico sono le seguenti:

```
gCursorDisplayList = glGenLists(1);
glNewList(gCursorDisplayList, GL_COMPILE);
qobj = gluNewQuadric();
gluQuadricDrawStyle(qobj, GLU_FILL);
gluQuadricNormals(qobj, GLU_SMOOTH);
gluQuadricOrientation(qobj, GLU_OUTSIDE);
gluCylinder(qobj, 0, 5, 15, 30, 1);
glTranslatef(0, 0, 15);
gluDisk(qobj, 0, 5, 30, 1);
gluDisk(qobj, 0, 1, 15, 1);
gluDeleteQuadric(qobj);
glEndList();
```

Dal momento che il proxy è rappresentato da un unico punto nello spazio, l'istruzione `glTranslatef(0, 0, 15)` permette di traslare il cono lungo l'asse *y* in modo tale che la sua punta coincida con la posizione del proxy.

Tramite i comandi:

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
 glEnable(GL_BLEND);
 glEnable(GL_POINT_SMOOTH);
 glHint(GL_POINT_SMOOTH_HINT, GL_NICEST);
 glEnable(GL_LINE_SMOOTH);
 glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
 glEnable(GL_POLYGON_SMOOTH);
 glHint(GL_POLYGON_SMOOTH_HINT, GL_NICEST);
```

contenuti all'interno della routine `initGL()` è stato aggiunto il supporto all'anti-aliasing; l'effettivo uso di tale tecnica di rendering dipende tuttavia dalle impostazioni dell'hardware grafico.

### 6.2.2 Rendering delle ombre

Una caratteristica grafica che si è voluta implementare per aumentare il realismo dello scenario senza aggiungere troppi dettagli è costituita dalle ombre; questa scelta è opportuna in quanto le ombre forniscono informazioni circa la posizione degli oggetti l'uno rispetto agli altri (e rispetto alla sorgente di luce, anche quando questa non viene rappresentata nella scena), ma non forniscono informazioni circa le proprietà aptiche, il materiale di cui è costituito l'oggetto o le sue tessiture più di quanto non faccia già la rappresentazione priva di ombre.

Un'ombra viene prodotta quando una fonte di luce colpisce un oggetto che oscura un altro oggetto o una superficie; i lati del primo oggetto che non vengono colpiti dalla luce vengono rappresentati con un colore più scuro, ma di default non viene proiettata nessuna ombra sugli altri oggetti o sulle superfici. Anche se esistono diversi modi per implementare questa caratteristica, la nostra attenzione si è focalizzata su due di questi.

### Trasposizione bidimensionale dell'oggetto

Con questo metodo, disegnare un'ombra è semplice: si crea una copia “appiattita” dell'oggetto e la si trasla secondo una matrice di trasposizione per farla giacere sullo stesso piano sul quale giace l'oggetto; la forma e la dimensione di questa trasposizione sono determinate dalla posizione della fonte di luce. Nelle librerie `glTools` [48] è contenuta la funzione `gltMakeShadowMatrix` che calcola la matrice di trasposizione sul piano bidimensionale; richiede in input un vettore di tre punti giacenti sul piano sul quale si vuole far apparire l'ombra, un vettore contenente la posizione della sorgente di luce e un puntatore alla matrice di trasformazione che deve essere creata.

```
void gltMakeShadowMatrix(GLTVector3 vPoints[3],  
                         GLTVector4 vLightPos,  
                         GLTMatrix destMat)  
{  
    GLTVector4 vPlaneEquation;  
    GLfloat dot;  
    gltGetPlaneEquation(vPoints[0], vPoints[1],  
                        vPoints[2], vPlaneEquation);  
  
    // Prodotto punto per punto dei vettori contenenti  
    // le posizioni del piano e della luce  
    dot = vPlaneEquation[0]*vLightPos[0] +  
          vPlaneEquation[1]*vLightPos[1] +  
          vPlaneEquation[2]*vLightPos[2] +  
          vPlaneEquation[3]*vLightPos[3];  
  
    // Calcolo della matrice di proiezione  
    // Prima colonna  
    destMat[0] = dot - vLightPos[0] * vPlaneEquation[0];  
    destMat[4] = 0.0f - vLightPos[0] * vPlaneEquation[1];  
    destMat[8] = 0.0f - vLightPos[0] * vPlaneEquation[2];  
    destMat[12] = 0.0f - vLightPos[0] * vPlaneEquation[3];
```

```
// Seconda colonna
destMat[1] = 0.0f - vLightPos[1] * vPlaneEquation[0];
destMat[5] = dot - vLightPos[1] * vPlaneEquation[1];
destMat[9] = 0.0f - vLightPos[1] * vPlaneEquation[2];
destMat[13] = 0.0f - vLightPos[1] * vPlaneEquation[3];

// Terza colonna
destMat[2] = 0.0f - vLightPos[2] * vPlaneEquation[0];
destMat[6] = 0.0f - vLightPos[2] * vPlaneEquation[1];
destMat[10] = dot - vLightPos[2] * vPlaneEquation[2];
destMat[14] = 0.0f - vLightPos[2] * vPlaneEquation[3];

// Quarta colonna
destMat[3] = 0.0f - vLightPos[3] * vPlaneEquation[0];
destMat[7] = 0.0f - vLightPos[3] * vPlaneEquation[1];
destMat[11] = 0.0f - vLightPos[3] * vPlaneEquation[2];
destMat[15] = dot - vLightPos[3] * vPlaneEquation[3];
}
```

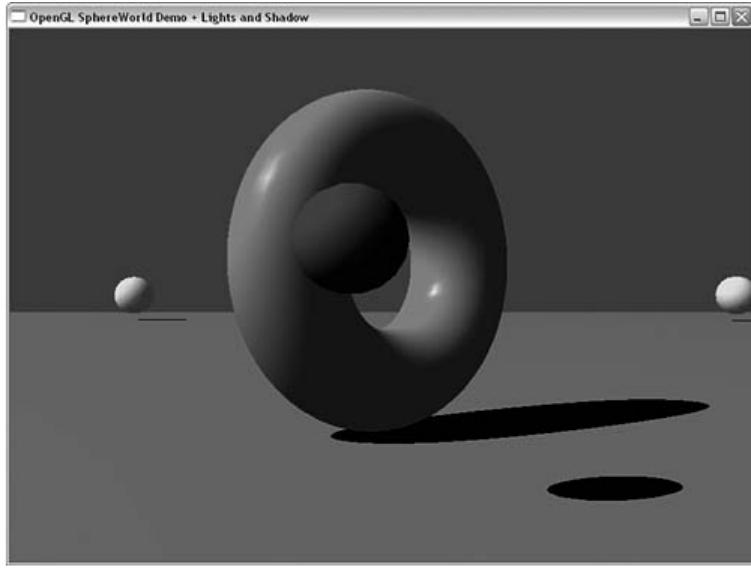
Moltiplicando la matrice ottenuta per la matrice della vista corrente, tutte le modifiche successive vengono trasposte sul piano. Nella nostra applicazione la sorgente di luce ha coordinate  $x$  e  $z$  nulle, quindi è stato necessario effettuare i calcoli relativi solo alla seconda colonna della matrice (infatti i calcoli relativi alle altre colonne restituiscono sempre 0).

Come si può ben capire, questa tecnica può essere utilizzata solo in rendering di scene nelle quali è presente un solo oggetto su un piano, oppure con più oggetti giacenti sullo stesso piano ma opportunamente spaziati (in modo che non ci si aspetti che un oggetto proietti la propria ombra su un altro), in quanto l'ombra viene proiettata solamente sul piano e non sugli altri poligoni.

## Shadow mapping

Questa tecnica è più complessa, ma l'idea su cui si basa è molto semplice: le zone in ombra sono quelle che non vengono colpite dalla luce. Se poniamo il nostro punto di vista nella stessa posizione in cui si trova la luce e guardiamo nella direzione in cui quest'ultima è diretta, vediamo tutto quello che dovrà essere illuminato; ciò che non vediamo è in ombra.

Effettuando il rendering della scena dal punto di vista della sorgente di luce, otteniamo un depth buffer contenente, per ogni pixel, le informazioni circa la distanza relativa tra questa sorgente e la superficie più vicina in una certa direzione; questa superficie è illuminata, tutte quelle che si trovano oltre restano nell'om-



**Figura 6.2:** Esempio di ombre ottenute tramite trasposizione bidimensionale degli oggetti

bra. L'algoritmo di shadow mapping consiste proprio nell'effettuare il rendering della scena dal punto di vista della sorgente di luce, copiare successivamente il contenuto del depth buffer in una texture, fare il rendering dal punto di vista della camera e applicare la nuova texture per determinare le zone di ombra.

Nell'applicazione Phantom Friction la tecnica di shadow mapping non viene utilizzata di default; per attivarla è sufficiente compilare il codice specificando la direttiva `_SHADOWMAPPING_` per il preprocessore. L'inizializzazione è contenuta nella routine `ShadowMappingInit()`, la quale comprende il caricamento delle texture che non devono essere modificate e l'abilitazione dell'estensione `GL_ARB_shadow` (se disponibile, questa estensione consente di eseguire un passaggio in meno nel calcolo delle ombre). La funzione di callback per il ridisegno della scena è:

```
void glutDisplay()
{
    ShadowMappingFirst(gCameraPosWC);
    drawScene();
    ShadowMappingSecond();
    RegenerateShadowMap(light0_position, shadowSize);
}
```

Per prima cosa vengono impostate le luci:

```
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
```

si effettua un confronto sulle ombre e si imposta il piano della visuale corrente:

```
glEnable(TEXTURETYPE);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glTexParameteri(TEXTURETYPE, GL_TEXTURE_COMPARE_MODE,
                GL_COMPARE_R_TO_TEXTURE);
glTexParameteri(TEXTURETYPE, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(TEXTURETYPE, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);
glEnable(GL_TEXTURE_GEN_Q);
glTexGenfv(GL_S, GL_EYE_PLANE, sPlane);
glTexGenfv(GL_T, GL_EYE_PLANE, tPlane);
glTexGenfv(GL_R, GL_EYE_PLANE, rPlane);
glTexGenfv(GL_Q, GL_EYE_PLANE, qPlane);
```

Tramite i comandi:

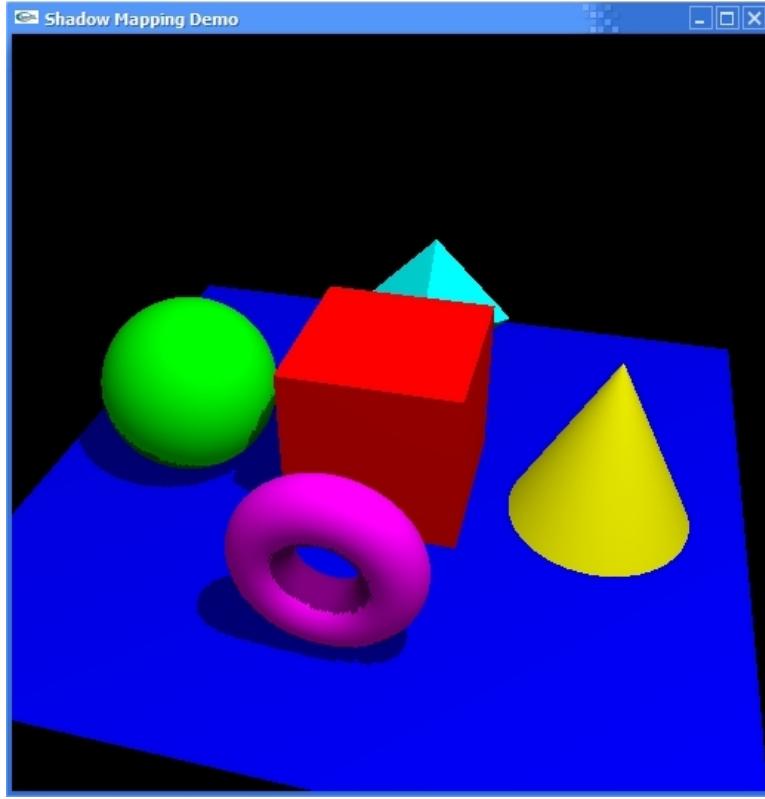
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(light[0], light[1], light[2],
          0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
glGetFloatv(GL_MODELVIEW_MATRIX, lightModelview);
glViewport(0, 0, shadowsize, shadowsize)
```

viene spostato il punto di vista in corrispondenza della sorgente di luce. Si copia il contenuto del depth buffer in una texture bidimensionale:

```
glCopyTexImage2D(TEXTURETYPE, 0, GL_DEPTH_COMPONENT,
                  0, 0, shadowsize, shadowsize, 0);
```

Al successivo rendering della scena si torna alla prospettiva originale con i comandi:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(40.0f, 1.0f, 274.0f, 1899.0f);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(cam[0], cam[1], cam[2], 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
glViewport(0, 0, windowHeight, windowHeight);
```



**Figura 6.3:** Esempio di applicazione dello shadow mapping.

### Scelta della tecnica di rendering delle ombre

Nell'applicazione Phantom Friction l'unico oggetto che deve proiettare un'ombra è il cursore, e la proietta su un piano; quindi la tecnica più adatta (e implementata di default) è la trasposizione bidimensionale. Effettuando questa scelta si è anche tenuto conto del fatto che questa tecnica è meno dispendiosa in termini di risorse di calcolo rispetto allo shadow mapping, e quindi è meno probabile che causi il verificarsi di latenze. Il cursore viene disegnato tramite il comando:

```
redrawCursor(true,0);
```

poi viene trasposto e disegnato una seconda volta, questa volta come ombra:

```
glMultMatrixf((GLfloat *)shadowMat);
redrawCursor(true,1);
```

L'unico inconveniente nel quale si incorre è che, se il cursore viene spostato oltre il limite della superficie, l'ombra viene disegnata lo stesso, anche se l'utente si aspetta di non vederla; per ovviare a questo problema sono state apportate due semplici modifiche:

- l'ombra viene disegnata solo se la posizione lungo l'asse  $y$  del proxy è maggiore o uguale della posizione del piano lungo lo stesso asse;

- l'ombra e lo sfondo hanno lo stesso colore: in tal modo, quando l'ombra esce dalla superficie, si confonde con lo sfondo e, per l'occhio umano, non è distinguibile da questo.

### 6.2.3 Aggiunta di texture grafiche

E' stata prevista la possibilità di aggiungere texture grafiche bidimensionali alle superfici degli oggetti, anche se ciò non rientra negli scopi dell'applicazione in quanto la percezione visiva di una tessitura distoglie l'utente dalle percezioni aptiche e uditive; tale caratteristica è disattivata per default, e per attivarla occorre procedere alla ricompilazione del codice sorgente includendo la stringa `_TEXTURE` tra le direttive del preprocessore.

La funzione:

```
loadTexture();
```

carica in memoria tutte le texture indicate nell'array `textureFiles`; è sufficiente effettuare il caricamento durante l'inizializzazione dell'applicazione, successivamente le texture resteranno disponibili in memoria fino alla chiusura dell'applicazione stessa. Il comando:

```
glBindTexture(GL_TEXTURE_2D, textures[n]);
```

è utilizzato per applicare effettivamente la texture in posizione *n* nell'array all'oggetto corrente.

## 6.3 Implementazione aptica

### 6.3.1 Rendering aptico degli oggetti virtuali

Per l'implementazione del rendering grafico si sono utilizzate sia le HD API che le HL API. Le HL API sono molto utili quando si vogliono impostare le proprietà aptiche degli oggetti virtuali e per gestire i frame; ad esempio, per effettuare il rendering aptico della superficie sono sufficienti le seguenti istruzioni:

```
hlBeginShape(HL_SHAPE_FEEDBACK_BUFFER, obj.shapeId);

hlMaterialf(HL_FRONT_AND_BACK, HL_STIFFNESS, obj.hap_stiffness);
hlMaterialf(HL_FRONT, HL_DAMPING, obj.hap_damping);
hlMaterialf(HL_FRONT, HL_STATIC_FRICTION, obj.hap_static_friction);
hlMaterialf(HL_FRONT, HL_DYNAMIC_FRICTION, obj.hap_dynamic_friction);
```

```
glCallList(obj.displayList);
```

```
hlEndShape();
```

con le quali vengono impostate la rigidità, il coefficiente di smorzamento, frizione statica e frizione dinamica. Le primitive grafiche utilizzate sono davvero poche, e ciò, come è stato discusso nel paragrafo 5.8.6, consiglia l'utilizzo del feedback buffer per il rendering aptico (e di conseguenza anche per il rendering grafico); questo viene impostato tramite il primo argomento di `hlBeginShape`.

Nella rappresentazione del cursore non devono essere impostate proprietà aptiche, ma è necessario calcolare le traslazioni e le trasformazioni del proxy:

```
hlGetDoublev(HL_PROXY_TRANSFORM, proxytransform);
glMultMatrixd(proxytransform);
```

Tutti i comandi riguardanti il rendering grafico della superficie, del cursore e delle ombre sono contenuti all'interno di un blocco `hlBeginFrame–hlEndFrame`:

```
hlBeginFrame();

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

// disegna la superficie
drawDraggableObjects(false,0);

glPushMatrix();
glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT0,GL_POSITION,lightPos);

// disegna il cursore
redrawCursor(true,0);

glPopMatrix();
glDisable(GL_DEPTH_TEST);
glDisable(GL_LIGHTING);
glPushMatrix();

// disegna l'ombra del cursore
hlGetDoublev(HL_PROXY_POSITION, posHD);
if(posHD[1] > minCursorY )
{
    glMultMatrixf((GLfloat *)shadowMat);
```

```
    redrawCursor(true,1);
}

// ripristina la normale prospettiva
glPopMatrix();

glEnable(GL_DEPTH_TEST);

hlEndFrame();
```

Sempre utilizzando le HLAPI è stata abilitata l'ottimizzazione *haptic camera view*:

```
hlEnable(HL_HAPTIC_CAMERA_VIEW);
```

in modo che venga effettuato il rendering aptico anche della porzione di superficie non visibile nella scena. Con l'istruzione:

```
hlHinti(HL_SHAPE_FEEDBACK_BUFFER_VERTICES, 100);
```

viene allocata memoria per 100 vertici, sufficienti per la rappresentazione in feedback buffer delle poche primitive usate; ciò si traduce in un risparmio di memoria in quanto di default viene riservata memoria sufficiente per 65536 vertici.

### 6.3.2 Rilevamento dei contatti e rendering delle tessiture aptiche

Tutte le istruzioni viste fino a questo punto sono sufficienti a fare in modo che, tramite l'uso del Phantom® Omni™, il parallelepipedo possa essere avvertito come un oggetto solido, senza la possibilità di penetrarlo. Tuttavia le proprietà aptiche restano costanti lungo tutta la superficie dell'oggetto; se si aggiunge il fatto che tale superficie è piana, si può intuire come questa sia costituita da una tessitura aptica uniforme su tutto l'oggetto. Durante la simulazione bimodale (cioè quando le simulazioni aptica e sonora sono eseguite contemporaneamente) si viene ad avvertire così una superficie che al tatto si presenta come costantemente ruvida (con livello di ruvidità proporzionale ai valori scelti di frizione statica e dinamica) accompagnata da suoni di sfregamento che descrivono una tessitura frattale e non costante. La discrepanza tra le due modalità non è avvertibile quando vengono simulate superfici lisce o poco ruvide, mentre si fa più evidente all'aumentare della ruvidità. Sono state valutate le seguenti metodologie per l'effettiva implementazione di una tessitura aptica.

## Modifica della geometria della superficie

La superficie viene avvertita apticamente come piana in quanto graficamente è piana. Un modo per far avvertire una superficie ruvida è modificarne la geometria, con l'aggiunta ad esempio di diversi poligoni a forma di piccoli coni che ne rappresentino le asperità; aggiungendo queste primitive alla lista contenente il parallelepipedo, tutti gli oggetti avranno le stesse proprietà aptiche e verranno rappresentati (sia dal punto di vista grafico che dal punto di vista aptico) come un unico oggetto. Disponendo i piccoli coni in modo opportuno, si ottiene una superficie che può essere avvertita come ruvida da entrambi i punti di vista aptico e grafico, e il livello di ruvidità dipende dalla dimensione, quantità e disposizione dei coni.

Tale soluzione però contrasta con i nostri scopi in quanto si dà un chiaro suggerimento visivo all'utente sulla tipologia di superficie, facendo passare totalmente in secondo piano le caratteristiche aptiche e sonore.

Si potrebbe pensare di rappresentare le asperità (cioè i piccoli coni) solo dal punto di vista aptico e non da quello grafico; dato che anche tutte le primitive vengono rappresentate come oggetti solidi (e quindi non penetrabili dalla sonda), con tale metodo si verrebbero a creare zone in cui il proxy si “blocca” senza urtare la porzione di superficie rappresentata graficamente (si verifica un urto aptico ma non un urto grafico), portando ad una notevole discrepanza tra le componenti aptica/grafica e creando di conseguenza solo confusione nell'utente.

## Modifica delle proprietà aptiche secondo pattern geometrici

Una seconda soluzione consiste nel modificare le proprietà aptiche della superficie lungo la superficie stessa, invece di mantenerle costanti. Il coefficiente di rigidità non può essere variato in quanto il parallelepipedo che rappresenta la superficie deve essere avvertito sempre come solido; la modifica del coefficiente della forza di smorzamento non influenza il rendering aptico in questa implementazione; quindi gli unici due parametri che possono essere variati sono le frizioni statica e dinamica. Entrambe sono forze che si oppongono al movimento tangenziale di un corpo.

Se pensiamo ad una superficie liscia come una superficie sulla quale i corpi strisciano senza incontrare resistenza, mentre pensiamo ad una superficie ruvida come una superficie sulla quale gli oggetti strisciano incontrando una forza di resistenza ogniqualvolta urtano un'asperità, allora vediamo come un aumento improvviso del coefficiente di frizione statica in alcuni punti può simulare la presenza di asperità in questi stessi punti, senza ricorrere a variazioni della geometria dell'oggetto.

In una prima implementazione si è scelto di mantenere i coefficienti delle due frizioni costanti e inferiori a 0.5 (ricordiamo che i coefficienti possono assumere valori compresi tra 0 e 1) nella simulazione di superfici lisce o poco lisce. Per le superfici ruvide i coefficienti vengono mantenuti ad un valore basso (0.2) e incrementati solo in alcuni punti, determinati in base alle coordinate geometriche del punto stesso: se il cursore si trova a contatto con la superficie e la sua posizione lungo l'asse  $x$  soddisfa la condizione:

```
(( (int)posHD[0]*10 ) % 2) != 0
```

(dove `posHD[0]` indica la posizione del proxy lungo l'asse  $x$ ) allora la frizione viene impostata ad un valore tanto più elevato quanto più ruvida deve risultare la superficie. Dal momento che si considera solo l'asse  $x$ , le asperità vengono posizionate lungo linee parallele all'asse  $z$  e giacenti sullo stesso piano della superficie (ovvero quello determinato dagli assi  $x$  e  $z$ ).

Se la precedente condizione viene sostituita dalla seguente:

```
(( (int)( sqrt(posHD[0]*posHD[0]+posHD[2]*posHD[2])*10 ) % 2) != 0
```

(dove `posHD[0]` indica ancora la posizione del proxy lungo l'asse  $x$ , mentre `posHD[2]` indica la posizione lungo l'asse  $z$ ), le asperità vengono posizionate secondo cerchi concentrici giacenti sul piano  $xz$ .

Possono essere implementati facilmente altri pattern geometrici; i micro-contatti dei quali viene simulato il suono emesso non sono però disposti secondo pattern regolari, ma casualmente secondo un pattern frattale. Rimane quindi una certa discrepanza tra le sensazioni aptiche e le sensazioni uditive per superfici non lisce.

### Modifica delle proprietà aptiche secondo pattern frattali

Il passo successivo consiste nella creazione di una tessitura aptica frattale.

Si potrebbe pensare di scrivere una funzione che implementi l'algoritmo discusso al paragrafo 3.1.2; ciò introdurrebbe un ulteriore appesantimento nel carico computazionale, senza garantire che il pattern generato nella patch per PD e quello implementato nell'applicazione varino concordemente. Il metodo più semplice e più efficiente risulta invece quello di utilizzare i dati calcolati nella subpatch `holy-roller~` (paragrafo 3.3.6). Il blocco `circ_max_filter~` calcola il profilo della superficie sulla quale avvengono i micro-contatti; il valore inviato in output viene letto dall'applicazione Phantom Friction (vedremo nella prossima sezione come avviene questa lettura) e, opportunamente scalato, viene utilizzato come coefficiente di proporzionalità per i valori di frizione statica e dinamica.

Nel paragrafo 4.4.2 sono state analizzate diverse tecniche di rendering delle tessiture. Quella qui implementata può essere descritta come una perturbazione delle forze (senza l'utilizzo della formula di Max e Becker); inoltre, in modo simile a quanto avviene per le tessiture aptiche basate sulle immagini, come indicatore della profondità della tessitura viene usato il valore del profilo della superficie calcolato nelle patch in Pure Data (descritte nel paragrafo 3.3.6).

Per implementare questo procedimento sono state utilizzate tre funzioni di callback. La prima rileva il verificarsi del primo contatto tra proxy e superficie e imposta a `true` un valore booleano:

```
void HLCALLBACK touchCallback(HLenum event, HLuint object,
                               HLenum thread, HLcache *cache,
                               void *userdata)
{
    touching = true;
}
```

La seconda invece imposta questo valore a `false` quando il proxy non è più in contatto con la superficie:

```
void HLCALLBACK untouchCallback(HLenum event, HLuint object,
                                 HLenum thread, HLcache *cache,
                                 void *userdata)
{
    touching = false;
}
```

Tale valore booleano viene usato dalla terza funzione di callback come condizione di abilitazione del calcolo (e del conseguente invio alla patch in PD) della velocità corrente e del fattore di amplificazione proporzionale alla forza normale.

```
HDCallbackCode HDCALLBACK velocitaCallback(void *pUserData)
{
    if (touching)
    {
        // richiesta della velocità corrente
        hdGetDoublev(HD_CURRENT_VELOCITY, veloHD);
        // richiesta della forza corrente
        hdGetDoublev(HD_CURRENT_FORCE, forceHD);

        velocita = sqrt( pow(veloHD[0],2) + pow(veloHD[2],2) );
```

```
// scrittura in memoria dei dati aptici dello scenario
setOpeData( &data,
    (float)( sqrt( abs(velocita * forceHD[1]) )/1000 ),
    velocita/1000,
    0,
    sceneDepth[indexScene],
    0,
    0,
    sceneNoiseFile[indexScene]);

WriteOpeData(&data);

pdSurface = ReadSurface();

hap_static_friction = ( pdSurface - a )/ ( b );
}

else
{
    setOpeData( &data, 0, sceneDepth[indexScene], 0, 0,
        0, 0, sceneNoiseFile[indexScene]);

    WriteOpeData(&data);
}

return HD_CALLBACK_CONTINUE;
}
```

Questa callback è stata implementata utilizzando le HD API come callback asincrona, in modo tale che venga eseguita non appena viene schedulata. Ciò è importante in quanto qui vengono generate le tessiture aptiche, e tale funzione deve essere eseguita alla velocità propria del rendering aptico, applicando l'effetto ad ogni iterazione del servo loop. L'istruzione

```
hap_static_friction = ( pdSurface - a )/ ( b );
```

imposta i coefficienti di frizione al valore ricevuto da PD (`pdSurface`) e opportunamente scalato nell'intervallo (`a,b`), dove `a` è impostato a 0 (si suppone che la superficie abbia sempre un valore positivo) e `b` corrisponde a 0.04 (calcolato empiricamente). La velocità viene calcolata come modulo delle componenti lungo gli assi `x` e `z` (il piano su cui giace la superficie) e viene divisa per 1000 per adeguarla alle unità di misura usate dalla patch `sliding.pd`: in quest'ultima

la velocità è assunta in *m/s*, mentre le HD API ritornano tale valore in *mm/s*. Tramite l'istruzione:

```
sqrt( abs(velocita * forceHD[1]) ),
```

dove `forceHD[1]` è la componente della forza esercitata dal dispositivo aptico lungo l'asse *y*, viene calcolato il fattore di proporzionalità dell'ampiezza del suono rispetto alla forza normale (vedi pagina 53).

## 6.4 Scambio dei dati tra l'applicazione Phantom Friction e le patch in Pure Data

Punto cruciale dello sviluppo di questa applicazione è stata l'implementazione di un'interfaccia di comunicazione con Pure Data, in modo da consentire lo scambio dei dati tra i due processi in tempo reale durante la simulazione bimodale. È stato importante curare questo aspetto perché la comunicazione dei dati deve avvenire alla stessa frequenza del rendering aptico, quindi 1000 volte al secondo; se le operazioni di scambio di informazioni svolte ad ogni ciclo sono troppo onerose, si rischia di introdurre latenze eccessive.

La simulazione bimodale ha significato se gli stimoli aptico, visivo e uditivo vengono percepiti dall'utente simultaneamente. Possiamo distinguere tre momenti ad ogni ciclo della simulazione: il momento in cui il proxy incontra una asperità, il momento in cui il dispositivo aptico rileva questo contatto e invia una forza in retroazione e il momento in cui viene generato il suono per questo evento. Tutti e tre i momenti elencati devono essere distanti (nel tempo) tra di loro il meno possibile, in accordo con la percezione dell'utente; se uno dei tre momenti è temporalmente troppo distante dagli altri (si verifica cioè una latenza), lo stimolo associato verrà percepito come estraneo all'evento. Se invece tutti e tre gli stimoli si verificano con latenze minori di certi livelli, allora verranno percepiti come contemporanei e l'utente li individuerà come componenti di uno stesso evento.

Si possono distinguere latenze *intramodali* e latenze *intermodali*:

**latenze intramodali** – le latenze intramodali si verificano tra stimoli dello stesso tipo: tra due stimoli sonori, tra due stimoli aptici o tra due stimoli visivi; i valori di latenza minimi percepibili variano a seconda della modalità considerata e corrispondono a 2 millisecondi per il suono, 27 millisecondi per il tatto e 43 millisecondi per la vista [27];

**latenze intermodali** – le latenze intermodali si verificano tra stimoli di tipo diverso (ad esempio si considera la latenza dello stimolo sonoro rispetto

agli stimoli visivo e aptico), e i valori minimi sono diversi a seconda che lo stimolo preceda o segua gli altri: se un suono precede gli stimoli visivo e aptico, la massima latenza accettata è di 25 millisecondi, mentre se li segue tale latenza sale a 42 millisecondi [27].

Se consideriamo che Pure Data, sintetizzando i suoni a 44100 Hz e avendo un buffer di 64 campioni, introduce una latenza di 1,45 millisecondi, mentre nell'applicazione Phantom Friction l'uso delle HLAPI introduce una latenza di 10 millisecondi (per le HDAPI la latenza è di 1 millisecondo), si vede che resta un margine di circa 15 millisecondi per una discrepanza temporale tra gli stimoli che sia accettabile.

Le soluzioni per la comunicazione tra i due processi sono essenzialmente tre [6].

### Protocollo di rete

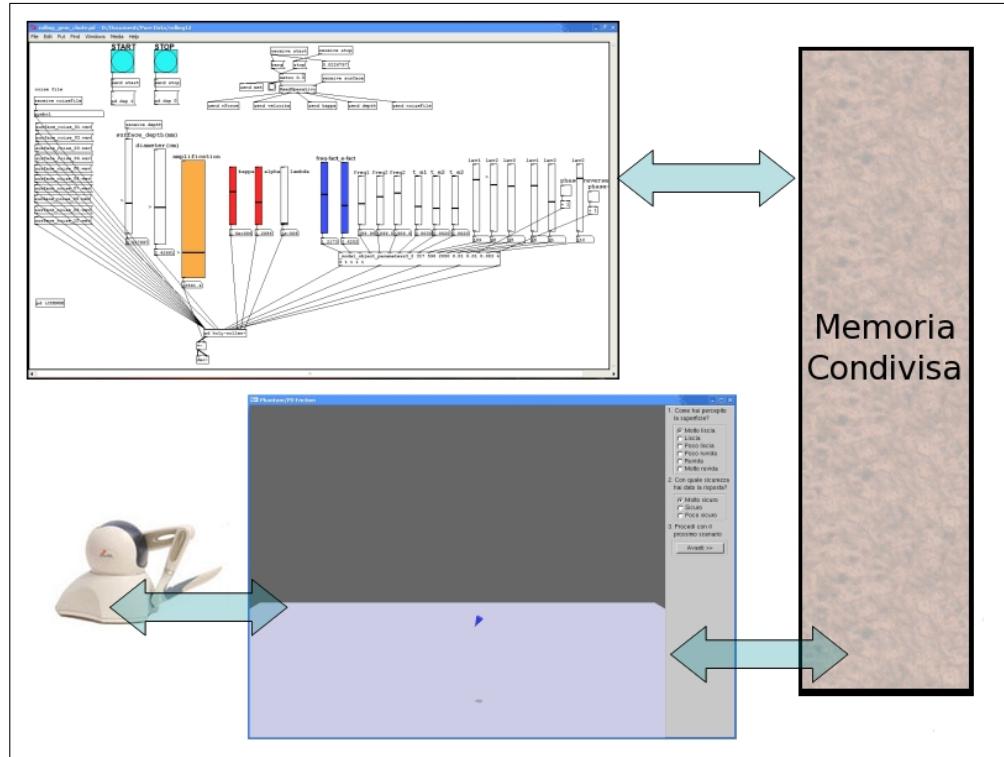
Dato che in Pure Data è supportata la lettura dei dati (via protocollo TCP e UDP) tramite l'oggetto `netreceive~`, sarebbe sufficiente implementare la lettura/scrittura dei dati nell'applicazione Phantom Friction tramite l'uso dei socket. Il vantaggio è che i due processi potrebbero così comunicare anche in remoto, facendo eseguire la simulazione aptica e quella sonora su due macchine diverse. Lo svantaggio è che la latenza dipende fortemente dalle caratteristiche dell'interfaccia di rete della macchina e dal traffico presente sulla rete. Inoltre viene ridotta la portabilità del codice, in quanto la programmazione dei socket in C++ è dipendente dal sistema operativo.

### Driver di periferica

Si potrebbe creare un external per Pure Data che legga i dati utilizzando i driver del dispositivo Phantom® Omni™. Questa soluzione è notevolmente complessa, a causa della scarsità di documentazione in merito e a causa del fatto che non si potrebbero estrarre i dati relativi alla geometria degli oggetti e agli istanti di collisione. Inoltre resterebbe irrisolto il problema di poter leggere i dati di Pure Data tramite l'applicazione Phantom Friction.

### Memoria condivisa

La tecnica più efficiente si è rivelata la predisposizione di un'area di memoria condivisa per la lettura/scrittura dei dati da parte di entrambi i processi. Lo svantaggio di questa soluzione è che dipende dalla piattaforma sulla quale il programma viene eseguito; inoltre Pure Data non possiede un oggetto interno che implementi una funzione utile allo scopo, perciò è stato necessario scrivere un



**Figura 6.4:** Schema di utilizzo di un'area di memoria condivisa per lo scambio dei dati tra il processo di sintesi aptica e quello di sintesi audio.

external. Il vantaggio maggiore (che compensa gli svantaggi citati) è la velocità con la quale avvengono le operazioni di lettura e scrittura.

In particolare è stata creata una struttura dati che contenga tutti i valori che dovranno essere trasferiti dall'applicazione Phantom Friction a Pure Data:

```
typedef struct
{
    double nForce;
    double velocita;
    float kappa;
    float depth;
    float statFr;
    float dynFr;
    int noiseFile;
} OpeData;
```

I valori che costituiscono la struttura `OpeData` sono i seguenti:

- `nForce` è la radice quadrata del prodotto tra forza normale e velocità;
- `velocita` indica la velocità corrente del proxy in  $m/s$ ;
- `kappa` è il coefficiente di elasticità della superficie;

- `depth` è il fattore di amplificazione del segnale di rumore frattale;
- `statFr` è il coefficiente di frizione statica;
- `dynFr` è il coefficiente di frizione dinamica;
- `noiseFile` è un numero intero che indica l'indice del file contenente il rumore frattale.

Nell'implementazione attuale solo i valori `nForce`, `velocita`, `depth` e `noiseFile` vengono effettivamente utilizzati. Infatti la patch in PD non ha bisogno di conoscere i coefficienti di frizione statica e dinamica, mentre il coefficiente di elasticità resta invariato per scelte implementative. E' possibile modificare i valori che fanno parte della struttura dati tramite la funzione:

```
void setOpeData(OpeData *data, double nForce,
                 double velocita, float kappa,
                 float depth, float statFr,
                 float dynFr, int noiseFile)
{
    data->nForce=nForce;
    data->velocita=velocita;
    data->kappa=kappa;
    data->depth=depth;
    data->statFr=statFr;
    data->dynFr=dynFr;
    data->noiseFile=noiseFile;
}
```

La gestione della memoria è implementata nel file `PhantomMemory.h`. Tramite la seguente funzione avviene la creazione dell'area di memoria condivisa tra i processi:

```
HANDLE hFile;

int CreatePhantomMemory()
{
    hFile = CreateFileMappingW(INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE,
                               0, sizeMem, (LPCWSTR)"PhantomMemory");
    if (hFile == NULL)
    {
        printf("ERROR: Unable to create a fileMapping.\n");
        return 1;
    }
```

```
hView = MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,0);

if (hView == NULL)
{
    printf("ERROR: Unable to map a viewOfFile.\n");
    return 2;
}
return 0;
}
```

E' importante fare attenzione al parametro `sizeMem`, il quale indica la dimensione dell'area di memoria condivisa e viene utilizzato anche per calcolare la posizione dei diversi dati in memoria; un valore errato di `sizeMem` porta alla scrittura e lettura di dati errati.

Il codice seguente si occupa dell'apertura e chiusura dell'area di memoria:

```
int OpenPhantomMemory(void)
{
    hFile = OpenFileMappingW(FILE_MAP_ALL_ACCESS, FALSE,
                           (LPCWSTR)"PhantomMemory");

    if (hFile == NULL)
    {
        printf("ERROR: Unable to open the FileMapping.\n");
        return 3;
    }

    hView = MapViewOfFile(hFile,FILE_MAP_ALL_ACCESS,0,0,0);

    if (hView == NULL)
    {
        printf("ERROR: Unable to open the FileMapping.\n");
        return 4;
    }
    hookOped = (OpeData*)(hView);
    hookSurface = (float*)(hookOped+1);
    return 0;
}

int ClosePhantomMemory(void)
{
    if (!UnmapViewOfFile(hView))
```

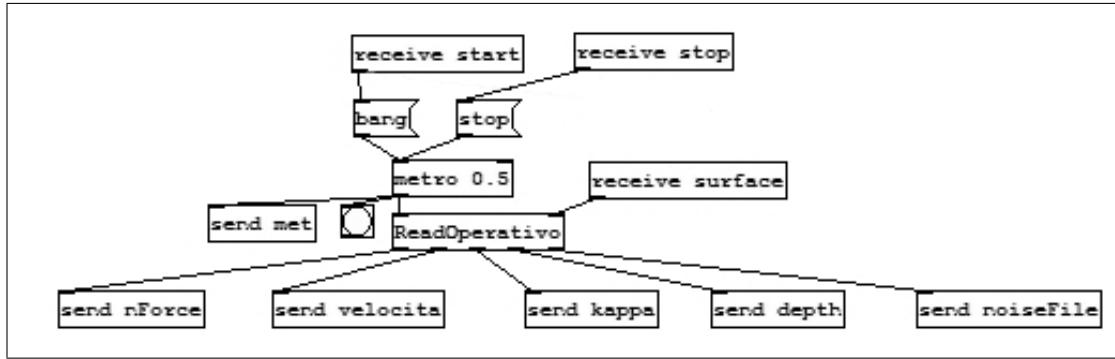
```
{  
    printf("ERROR: Could not unmap viewOfFile.\n");  
    return 5;  
}  
CloseHandle(hFile);  
return 0;  
}
```

Un oggetto `OpeData` può essere scritto in memoria e letto dalla memoria usando le due funzioni `WriteOpeData(OpeData *elemento)` e `ReadOpeData()` (nella prima deve essere passato come parametro un puntatore all'oggetto):

```
void WriteOpeData(OpeData *elemento)  
{  
    hookOped->nForce = elemento->nForce;  
    hookOped->velocita = elemento->velocita;  
    hookOped->kappa = elemento->kappa;  
    hookOped->depth = elemento->depth;  
    hookOped->statFr = elemento->statFr;  
    hookOped->dynFr = elemento->dynFr;  
    hookOped->noiseFile = elemento->noiseFile;  
}  
  
OpeData ReadOpeData()  
{  
    return *hookOped;  
}
```

Con semplici modifiche a questi due comandi si implementano la scrittura e lettura del valore della superficie generata dall'oggetto `circ_max_exp~`:

```
void WriteSurface(float *elemento)  
{  
    *hookSurface = *elemento;  
}  
  
float ReadSurface()  
{  
    return *hookSurface;  
}
```



**Figura 6.5:** L'oggetto ReadOperativo e le sue connessioni.

#### 6.4.1 La scrittura e la lettura della memoria da parte dell'applicazione Phantom Friction

E' l'applicazione Phantom Friction che si occupa di creare l'area di memoria condivisa, e lo fa non appena viene avviata; subito dopo la memoria viene aperta e si crea un oggetto `data` di tipo `OpeData` contenente tutti i valori impostati a 0, il quale viene scritto assieme ad un valore nullo di `pdSurface` allo scopo di effettuare un'inizializzazione della memoria stessa, cancellando eventuali dati presenti. All'interno della callback asincrona `velocitaCallback`, dopo aver calcolato i valori necessari alla impostazione corretta dell'oggetto `data`, avvengono in sequenza le seguenti operazioni:

- viene scritto in memoria l'oggetto `data`;
- viene letto dalla memoria il valore `pdSurface`;
- vengono calcolati i coefficienti di frizione statica e dinamica correnti.

Si è scelto di non fare un'apertura e una chiusura della memoria per ogni ciclo di queste operazioni, in primo luogo per evitare conflitti con le operazioni di lettura e scrittura svolte dalla patch in Pure Data; in secondo luogo ciò comporterebbe un overhead considerevole nel carico computazionale; basti pensare che, su un PC dotato di processore Intel® Core™ Duo T2400 con 2GB di RAM, un milione di cicli lettura/scrittura con apertura/chiusura della memoria richiede tra i 12 e i 13 secondi, mentre eliminando l'apertura/chiusura della memoria si scende a tempi dell'ordine di pochi millisecondi.

#### 6.4.2 La scrittura e la lettura della memoria da parte delle patch in Pure Data

Per consentire alle patch in Pure Data di leggere e scrivere dati in RAM è stato necessario scrivere un external; per semplicità ci si è appoggiati alla libreria

*flex* discussa nel paragrafo 3.3.4. Tale oggetto è stato chiamato `ReadOperativo` (figura 6.5); non necessita di argomenti di costruzione e possiede 2 inlet e 5 outlet: i due inlet ricevono rispettivamente un bang e il valore della superficie, mentre vengono mandati in output, in ordine:

- la radice quadrata del prodotto tra la velocità e la forza normale;
- la velocità corrente del proxy;
- il coefficiente di elasticità;
- il fattore di amplificazione del rumore frattale;
- il nome del file contenente il rumore frattale.

Tutto ciò viene inizializzato tramite il costruttore:

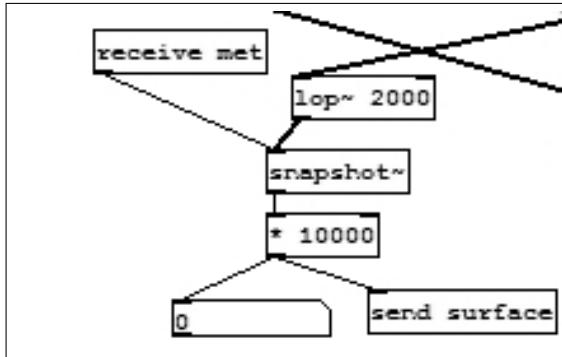
```
ReadOperativo::ReadOperativo()
{
    // aggiunta degli inlet
    AddInAnything("bang");//(0)
    AddInFloat("surface");//(1)

    // aggiunta degli outlet
    AddOutFloat("nForce");//(0)
    AddOutFloat("Velocita'");//(1)
    AddOutFloat("Kappa");//(2)
    AddOutFloat("Depth");//(3)
    AddOutSymbol("NoiseFile");//(4)

    // inizializzazione delle variabili
    lastResult = 0;
    lastDepth = 0;
    fileName = "";
    open = false;

    // registrazione dei metodi
    FLEXT_ADDBANG(0, ope_bang);
    FLEXT_ADDMETHOD(1, writeSurface);
}
```

Un oggetto `metro` invia un bang all'oggetto `ReadOperativo` ogni 0.5 millisecondi. Al primo bang ricevuto avviene l'apertura dell'area di memoria condivisa.



**Figura 6.6:** Filtraggio passa–basso del valore associato al profilo della superficie frattale.

Un ciclo di lettura e scrittura avviene ogni volta che viene ricevuto un bang, quindi la frequenza con la quale avvengono queste operazioni è di 2 KHz. Ai primi due outlet i valori vengono inviati aggiornati ad ogni ciclo, mentre i restanti vengono inviati solo quando vengono modificati; ciò è in accordo con il fatto che i primi due valori sono misure istantanee, mentre gli altri sono parametri che riguardano uno scenario e restano invariati all'interno dello stesso scenario.

Il valore della superficie non viene scritto direttamente in memoria, ma viene prima filtrato attraverso un filtro passa–basso a 2 KHz (figura 6.6), in modo da evitare il verificarsi di aliasing.

Infine il distruttore si occupa della chiusura della memoria condivisa, che pertanto viene effettivamente chiusa solo alla chiusura della patch `sliding.pd` (e solo se era stata aperta):

```

ReadOperativo::~ReadOperativo()
{
    if(open)
        ClosePhantomMemory();
}

```

### 6.4.3 Analisi delle prestazioni

Grazie alle scelte implementative fatte, si è giunti ad un'applicativo che consente una simulazione bimodale realistica. Le latenze sono ridotte entro i limiti mediamente accettabili dall'uomo, quindi l'utente percepisce gli stimoli visivo, aptico e sonoro come stimoli coerenti. Le prove sono state effettuate con un PC notebook dotato di processore Intel® Core™ Duo T2400 e 2 GByte di memoria RAM; tuttavia la scheda audio è di fascia medio–bassa, e nonostante ciò le prestazioni restano soddisfacenti. Utilizzando un hardware audio di fascia alta

sicuramente i valori di latenza degli stimoli sonori verrebbero ridotti al minimo, permettendo l'uso della simulazione anche su PC meno potenti.

# Capitolo 7

## Esperimenti sulla percezione audio-aptica

### 7.1 Introduzione

L'applicazione, completa nelle sue componenti di simulazione aptica e audio, è stata utilizzata per effettuare degli esperimenti di percezione audio-aptica. Questo ha due obiettivi: il primo è verificare la correttezza dell'applicazione, cioè serve a capire se effettivamente riesce ad infondere nei soggetti le sensazioni che noi vogliamo simulare, in particolare quella di controllare un corpo che striscia su una superficie; il secondo obiettivo è capire come avviene nei singoli soggetti la percezione bimodale audio-aptica, quale senso prevale tra tatto, vista e udito, quali caratteristiche della scena virtuale vengono prese in considerazione e quali invece vengono trascurate.

La percezione visiva non rientra nell'oggetto del nostro studio, infatti per questo motivo la scena presentata è graficamente molto semplice, l'essenziale per permettere una corretta interazione tramite l'uso del Phantom® Omni™; tuttavia in alcuni casi, come vedremo nel seguito, alcuni soggetti sono stati influenzati lo stesso nelle loro percezioni dalla componente grafica.

Il tipo di percezione uditiva che si vuole studiare riguarda l'ascolto dei suoni di tutti i giorni; in particolare non si vuole che l'utente riconosca la sorgente del suono, bensì le caratteristiche della sorgente (la sua ruvidità) e dell'evento che genera il suono (un rotolamento o uno sfregamento).

### 7.2 Studi precedenti

Sebbene gli esperimenti di percezione vengano svolti da più di trent'anni, solo negli ultimi tempi la tecnologia ha reso possibile l'utilizzo di strumenti che

permettano l'interazione con un ambiente virtuale. Quando tali dispositivi non erano disponibili, gli esperimenti di percezione aptica venivano condotti facendo interagire i soggetti con oggetti reali usando direttamente le mani; i suoni erano reali e non simulati; se non si volevano fornire suggerimenti visivi, l'oggetto veniva nascosto.

### 7.2.1 Lederman – Percezione uditiva delle tessiture (1978)

Susan Lederman [21] nel 1978 ha svolto un insieme di tre studi mirati a investigare il ruolo del suono prodotto dal tocco nella percezione della tessitura di una superficie, nei quali ai soggetti era data la possibilità di giudicare la ruvidità della superficie solamente basandosi sul suono. Questi giudizi si sono dimostrati simili, ma non uguali a quelli dati solo in base alle sensazioni aptiche. Negli esperimenti in cui venivano forniti entrambi i tipi di sensazioni, i soggetti tendevano ad usare maggiormente le informazioni aptiche.

#### Esperimento 1

Nel primo esperimento i soggetti devono stimare la ruvidità di un insieme di piatti di diversa dimensione basandosi solamente sul suono prodotto dal tocco di questi; il suono viene generato dallo sperimentatore. Dai risultati è emerso che la ruvidità percepita decresce leggermente all'aumentare della grandezza del piatto, mentre è proporzionale alla pressione esercitata con le dita, e tale effetto è più evidente quanto più grande è il piatto.

#### Esperimento 2

Nel secondo esperimento viene chiesto ai soggetti di giudicare la ruvidità degli stessi piatti, ma in due condizioni diverse. Nella prima la stima viene basata solo sulla percezione aptica (il suono viene mascherato); nella seconda invece ogni utente ha a disposizione sia gli stimoli aptici che gli stimoli uditivi, entrambi prodotti dall'utente stesso nella sua esplorazione dei piatti.

Non si sono verificate differenze significative tra i risultati nelle due modalità, e gli effetti riscontrati sono simili a quelli del primo esperimento, a differenza del fatto che la dimensione influisce ancora di più sulla ruvidità percepita, mentre la pressione esercitata ha un effetto minore. Rispetto al primo esperimento invece si è notato che la diminuzione della ruvidità percepita è più marcata quando vengono avvertite solo le sensazioni aptiche rispetto a quando vengono avvertite solo quelle uditive. Dato che i risultati delle due modalità (solo tatto e tatto più udito) non sono molto diversi, si deduce che in questo secondo esperimento gli

utenti non hanno usato molto le informazioni sul suono prodotto per discriminare le diverse ruvidità.

### Esperimento 3

In questo terzo esperimento sono stati usati piatti di ruvidità variabile, e i soggetti dovevano giudicare in tre diverse condizioni: solamente ascoltando il suono prodotto dallo sperimentatore, solamente esplorando con le dita la superficie dell'oggetto e infine avendo a disposizione entrambi i tipi di sensazioni.

La ruvidità percepita tende a crescere monotonicamente con l'aumentare dell'ampiezza delle scanalature sulla superficie dei piatti; quando sono presenti entrambi gli stimoli, i soggetti tendono a dare giudizi simili a quelli dati utilizzando solo il tatto, mentre la superficie viene giudicata più ruvida quando è presente solo lo stimolo uditivo. Sono stati analizzati anche gli effetti della forza applicata dalle dita e la velocità di movimento delle stesse sulla superficie durante l'esplorazione: la ruvidità stimata è tanto maggiore quanto maggiore è la forza applicata, e ciò indipendentemente dalle informazioni sensoriali coinvolte. Se si analizzano le differenze nelle stime effettuate applicando una maggiore o minore forza con le dita, si nota come queste siano più evidenti quando è presente solo lo stimolo uditivo e le dita vengono mosse lentamente; se il movimento è veloce, le differenze nelle stime sono più o meno le stesse in tutte le tre modalità.

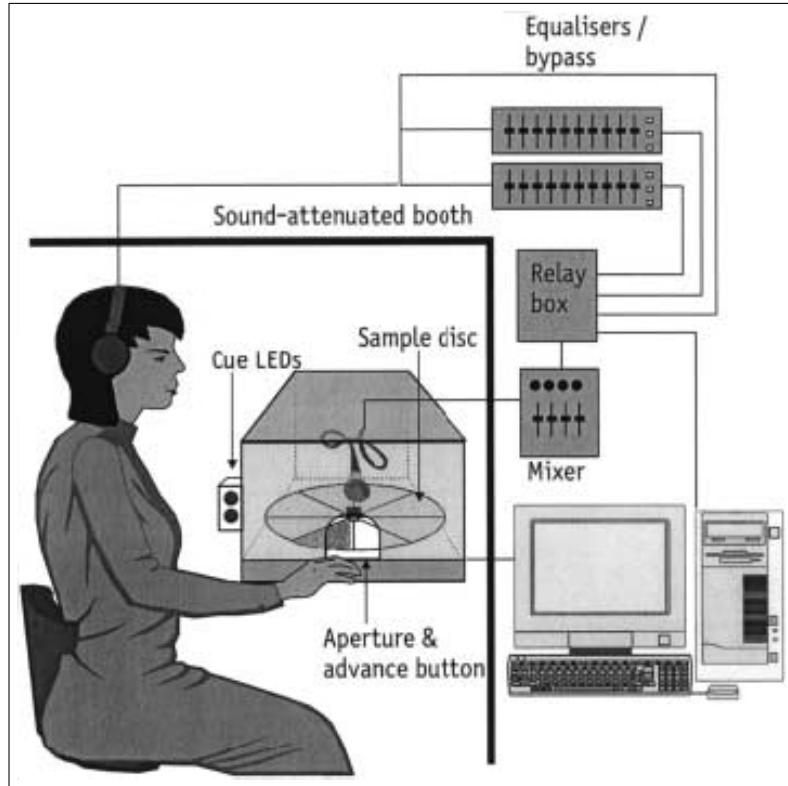
### Osservazioni generali

Considerando i risultati di tutti e tre gli esperimenti, i giudizi dei soggetti riflettono la loro abilità nel riconoscere i cambiamenti dovuti alle alterazioni di dimensioni, scanalature, pressione e velocità di movimento. Molto probabilmente i due aspetti degli stimoli acustici che hanno influito di più sulle stime sono l'altezza (frequenza fondamentale) e l'ampiezza; è stato notato infatti che l'altezza del suono tende a decrescere all'aumentare della profondità delle scanalature e della forza esercitata con le dita. Infine l'aumento della velocità di esplorazione provoca un aumento dell'altezza e dell'ampiezza.

#### 7.2.2 Guest, Catmur e Lloyd – Interazioni audio–aptiche nella percezione della ruvidità (2002)

Negli studi di Guest, Catmur e Lloyd [15] si è posta l'attenzione sulla cosiddetta *illusione pelle–pergamena*, cercando di dimostrarla attraverso tre esperimenti.

L'attrezzatura sperimentale prevede l'applicazione di fogli di carta abrasiva su un disco di plastica, il quale ruota grazie ad un motore. Tutto ciò è contenuto



**Figura 7.1:** Setup degli esperimenti di Guest, Catmur e Lloyd [15]: il soggetto è seduto di fronte alle attrezzature; l'apertura per la mano è stata coperta per nascondere il disco alla vista.

all'interno di una scatola di legno sulla quale è prevista un'apertura per poter inserirvi la mano, in modo da nascondere alla vista ciò che si tocca. Premendo un apposito pulsante posto sulla scatola, l'utente può avanzare di esperimento in esperimento. Il suono viene catturato da un microfono, processato in modo diverso a seconda del tipo di esperimento e inviato al soggetto tramite un paio di cuffie.

### Esperimento 1

Ai soggetti è stato chiesto di riconoscere ripetutamente quale tra due campioni è quello ruvido e quale quello liscio, trascurando il suono e basandosi esclusivamente sulle sensazioni date dal tocco della carta; tuttavia il suono prodotto viene modificato effettuando un'amplificazione o un'attenuazione in frequenza. Sono state proprio queste modifiche a cambiare le stime dei soggetti. In particolare, per il campione di carta liscia, amplificare le alte frequenze ha prodotto un forte incremento di stime errate rispetto all'attenuazione delle stesse frequenze; per il campione ruvido invece accade il contrario: amplificare le alte frequenze porta a meno errori nelle stime rispetto a quando le frequenze vengono attenuate.

## Esperimento 2

In questo caso il microfono cattura il suono prodotto mentre il soggetto si sfrega le mani; questo dovrà poi giudicare, oltre al livello di ruvidità, se il suono dà una sensazione di mani asciutte o umide.

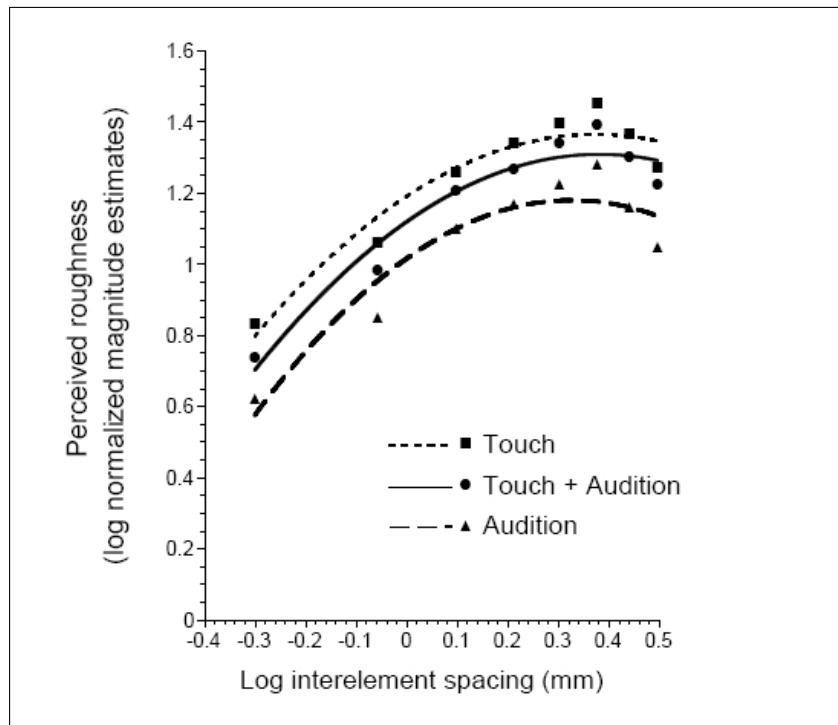
Per quanto riguarda l'analisi della ruvidità, la manipolazione dell'ampiezza del suono o delle frequenze non porta a cambiamenti significativi nelle stime. Quest'ultime invece cambiano quando entrambi gli effetti vengono applicati simultaneamente; in particolare, quando il livello di attenuazione è di -40dB, viene percepita una ruvidità via via maggiore al crescere delle amplificazioni in frequenza. Per l'analisi della ruvidità invece si ha che un suono più forte porta a percepire le mani come più asciutte, e lo stesso effetto si ha con un'amplificazione in alta frequenza.

## Esperimento 3

Rispetto al secondo esperimento, si è introdotta l'analisi degli effetti provocati da un ritardo applicato al suono percepito. E' risultato che con un ritardo di 150 e 300 millisecondi viene percepita una ruvidità maggiore rispetto a quando l'audio risulta perfettamente sincronizzato, mentre si ha la sensazione di mani meno asciutte. In entrambi i casi però l'effetto è più marcato quando il ritardo corrisponde a 150 millisecondi; il fatto che le percezioni non cambiano linearmente con l'aumentare del ritardo può essere spiegato dal fatto che lo sfregamento delle mani è un movimento periodico, e un ritardo di circa 300 millisecondi potrebbe riportare l'audio in fase con l'azione compiuta dal soggetto.

### 7.2.3 Lederman, Klatzky, Morgan e Hamilton – Integrazione delle informazioni multimodali sulla tessitura di una superficie tramite una sonda (2002)

In uno studio del 1986 [26], Lederman ha analizzato le relazioni che intercorrono nella percezione della ruvidità tra le sensazioni tattili e quelle visive. Agli utenti è stato chiesto di giudicare l'aspetto della superficie mediante il tocco, e la sensazione tattile che potrebbe fornire solo utilizzando la vista; nel 70% dei casi è stato il tatto a fornire maggiori informazioni sulla natura della superficie. Quando invece è stato chiesto di giudicare la densità spaziale della stessa, è stato il senso della vista a fornire più informazioni nel 70% dei casi. Ciò ha dimostrato che la risoluzione spaziale del sistema visivo umano è maggiore rispetto a quella del sistema cutaneo, mentre quest'ultimo è più preciso nella stima della ruvidità. Successivamente invece è stata presa in considerazione l'interazione tra tatto e



**Figura 7.2:** Andamenti delle stime della ruvidità come funzione dello spazio tra gli elementi in tre modalità di analisi (solo tatto, solo udito e tatto+uditio) .

udito [22]; contrariamente alle ricerche già citate, questa volta la superficie veniva esplorata usando una sonda in plastica rigida al posto delle dita. Dato che il suono generato dal contatto tra una superficie e una sonda rigida è più forte del suono generato quando sono le dita ad entrare in contatto con la stessa superficie, ci si potrebbe aspettare che le informazioni sonore giochino un ruolo più importante nella percezione bimodale. Ai soggetti è stato chiesto di giudicare la ruvidità di una superficie in tre distinte modalità: solo tramite il tatto, solo tramite l'udito e infine sfruttando entrambi i sensi.

Le stime maggiori sono state ottenute quando si poteva utilizzare solo il tatto, quelle minori invece quando era disponibile solo lo stimolo acustico (come è possibile vedere in figura 7.2); è stato calcolato che il peso delle informazioni tattili è del 62%, rispetto ad un 38% di quelle acustiche, anche se queste percentuali variano da soggetto a soggetto: in alcuni si è verificata una dominanza del senso del tatto, in pochi altri la dominanza dell'udito.

Variando la distanza tra le asperità della superficie, è stato notato che la ruvidità percepita cresce al crescere della distanza variata fino ad un certo valore; distanze oltre a questo valore portano ad un decremento della ruvidità percepita. In modo simile, al crescere dell'intervallo tra le asperità diminuisce la sicurezza con la quale i soggetti hanno compiuto le loro stime.

### 7.3 Aspetti innovativi

Il nostro esperimento è stato strutturato traendo spunto da tutte queste ricerche e basandosi in modo particolare su quelle di Susan Lederman. E' stata ripresa l'idea di analizzare la percezione audio-aptica in diverse modalità, anche se in una concezione diversa. Se con gli esperimenti precedenti ai soggetti venivano presentati prima solo gli stimoli aptici, poi solo quelli uditivi e infine entrambi, nel nostro caso vengono sempre presentati tutti gli stimoli contemporaneamente. Ciò che cambia è che nella prima serie di esperimenti gli stimoli audio e aptico variano concordemente, mentre nelle successive viene variato solo uno dei due stimoli a seconda della superficie esplorata, mentre l'altro viene mantenuto costante; in tal modo si creano le condizioni per investigare come e quanto i sensi del tatto e dell'udito si influenzano l'un l'altro. E' stato possibile realizzare ciò grazie al fatto che, rispetto agli studi precedenti, la superficie che il soggetto deve analizzare non è una superficie reale, bensì una rappresentazione virtuale tridimensionale di una superficie. Inoltre tale rappresentazione resta immutata al variare della ruvidità simulata, rendendo così possibile mascherare importanti informazioni visive senza mascherare la superficie stessa. Mascherare l'intera superficie avrebbe sicuramente creato problemi agli utenti durante l'esplorazione; infatti l'esplorazione "cieca" di un oggetto con l'uso delle mani è una cosa naturale, acquisita nelle esperienze di vita, ma l'uso di un dispositivo aptico è nuovo alla maggior parte degli utenti, così come l'esplorazione di un ambiente virtuale. Si capisce così come sia utile al soggetto avere un riferimento visivo per effettuare un'interazione corretta con gli oggetti simulati.

L'uso di un dispositivo aptico come sonda di esplorazione e oggetti virtuali al posto di oggetti reali introduce un'importante differenza rispetto agli esperimenti effettuati in passato: infatti non si può parlare di stimoli del senso del tatto negli stessi termini usati nella descrizione delle prove fatte usando l'esplorazione con le dita. Il sistema cutaneo e il dispositivo aptico hanno delle risoluzioni diverse; sicuramente il primo ha una risoluzione maggiore di quella che può fornire il secondo. Inoltre la nostra pelle è sensibile a pressioni, vibrazioni e deformazioni locali, ed è da tutti questi eventi che traiamo informazioni circa la natura dell'oggetto che tocchiamo. Il dispositivo aptico può trasmettere solo forze e vibrazioni, non deformazioni locali, rendendo quindi più difficile la discriminazione di tessiture a trame fitte; inoltre tale discriminazione è più difficile quanto più bassa è la risoluzione del dispositivo, implicando che le sensazioni trasmesse dipendono dal dispositivo utilizzato e dalla qualità dell'implementazione software della simulazione.

Queste ultime considerazioni possono essere applicate anche agli stimoli uditivi; anche in questo caso infatti lo stimolo trasmesso non è prodotto da un'intera-

zione reale, ma è simulato, e risulta tanto più verosimile quanto più accurati sono i modelli fisici usati e quanto più efficiente è la loro implementazione. Tuttavia ciò introduce una notevole flessibilità rispetto all'uso di campioni audio preregistrati; questi ultimi infatti devono subire complesse elaborazioni se vogliono essere usati per simulare materiali o caratteristiche diverse di uno stesso oggetto, mentre con l'uso dei modelli fisici è sufficiente variare pochi parametri per ottenere suoni notevolmente diversi.

L'uso di un ambiente virtuale porta ad un ulteriore elemento nuovo rispetto agli esperimenti precedenti: proprio perché quasi tutti i soggetti sono nuovi a tale tipo di esplorazione (e quelli che non sono nuovi hanno avuto solo poche esperienze occasionali), la memoria di esperienze passate viene a giocare un ruolo di minore importanza.

## 7.4 Attrezzature e organizzazione sperimentale

Gli esperimenti sono stati eseguiti utilizzando un notebook dotato di processore Intel® Core™ Duo T2400, 2 gigabyte di memoria RAM e acceleratore grafico ATI® .

Il soggetto è seduto di fronte al notebook, con il dispositivo Phantom® Omni™ alla sua destra; tramite quest'ultimo viene trasmesso il feedback aptico, mentre il feedback sonoro è trasmesso attraverso un paio di cuffie stereo.

L'interfaccia grafica dell'applicazione è come in figura 6.1; il rettangolo più chiaro rappresenta la superficie da esplorare, mentre il piccolo cono azzurro rappresenta l'estremità del dispositivo aptico. Il pannello sulla destra contiene due insiemi di pulsanti radio (cioè mutuamente esclusivi): il primo gruppo costituisce una scala divisa in sei gradazioni (da molto liscia a molto ruvida) per la stima della ruvidità di una superficie, mentre nel secondo gruppo è possibile scegliere il grado di sicurezza relativa alla scelta della ruvidità.

Ad ogni soggetto è stato illustrato il funzionamento generico del dispositivo aptico e come interagire con l'applicazione; non è stato svelato tuttavia lo scopo dell'esperimento, il dettagli dell'implementazione e della variazione dei parametri. Ad ogni esperimento, l'utente ha avuto la possibilità di esplorare la superficie virtuale per il tempo che credeva necessario; terminata l'esplorazione doveva scegliere dal pannello a destra un grado per la ruvidità percepita e il grado di sicurezza della risposta. Successivamente, cliccando sul pulsante **Avanti** si passa allo scenario successivo: graficamente non varia nulla, ma vengono variate le proprietà aptiche della superficie e i parametri di generazione del suono. Per fare in modo che l'utente si renda conto dell'effettivo cambiamento, è stato aggiunto un contatore autoincrementante che riporta l'indice della scena corrente.

I pulsanti radio vengono selezionati tramite l'uso del mouse o del touchpad; è stata implementata anche la possibilità di effettuare la selezione tramite il dispositivo aptico: infatti, nel momento in cui viene spostato il cursore al di fuori dello scenario tridimensionale posizionandolo in un altro punto della finestra, su un'altra finestra o sul desktop, si va a controllare il cursore del mouse. Tuttavia tale funzione è stata poi disabilitata in quanto sembrava indurre confusione negli utenti.

Ogni esperimento prevede l'esplorazione di un totale di 36 scenari: 9 sono gli scenari distinti, ed ognuno viene ripresentato 4 volte. I primi 9 scenari costituiscono una fase di *training implicito*: tutti i diversi scenari vengono presentati in ordine di ruvidità crescente, e sono utili al soggetto per prendere confidenza con il dispositivo aptico e con l'applicazione; di conseguenza, le percezioni relative a questi scenari non sono considerate, anche se l'utente non è consapevole di ciò. Successivamente vengono presentate tre serie composte dagli stessi scenari in ordine sparso.

I nove stimoli sono stati scelti empiricamente, valutando nove diverse combinazioni tra l'esponente  $\beta$  del rumore  $1/f^\beta$  e il fattore di amplificazione tali da produrre una sensazione di ruvidità crescente:

**scenario 1:**  $\beta = 0.2$ , fattore di amplificazione = 0.0010;

**scenario 2:**  $\beta = 1.2$ , fattore di amplificazione = 0.0012;

**scenario 3:**  $\beta = 2.0$ , fattore di amplificazione = 0.0030;

**scenario 4:**  $\beta = 0.6$ , fattore di amplificazione = 0.0080;

**scenario 5:**  $\beta = 1.4$ , fattore di amplificazione = 0.0100;

**scenario 6:**  $\beta = 1.8$ , fattore di amplificazione = 0.0120;

**scenario 7:**  $\beta = 1.6$ , fattore di amplificazione = 0.0160;

**scenario 8:**  $\beta = 1.8$ , fattore di amplificazione = 0.0180;

**scenario 9:**  $\beta = 2.0$ , fattore di amplificazione = 0.0210;

Per ogni soggetto e per ogni scenario distinto, i tre valori di ruvidità percepita raccolti sono stati sottoposti ad un'elaborazione preliminare, prima di essere utilizzati per una loro rappresentazione grafica. Per prima cosa è stata fatta una media di questi tre valori; successivamente i valori medi sono stati normalizzati, dividendo ognuno di questi valori per la media dei valori percepiti dal soggetto e moltiplicando per la media complessiva relativa a tutti i soggetti. Sulle stime così ottenute è stata svolta una analisi ANOVA.

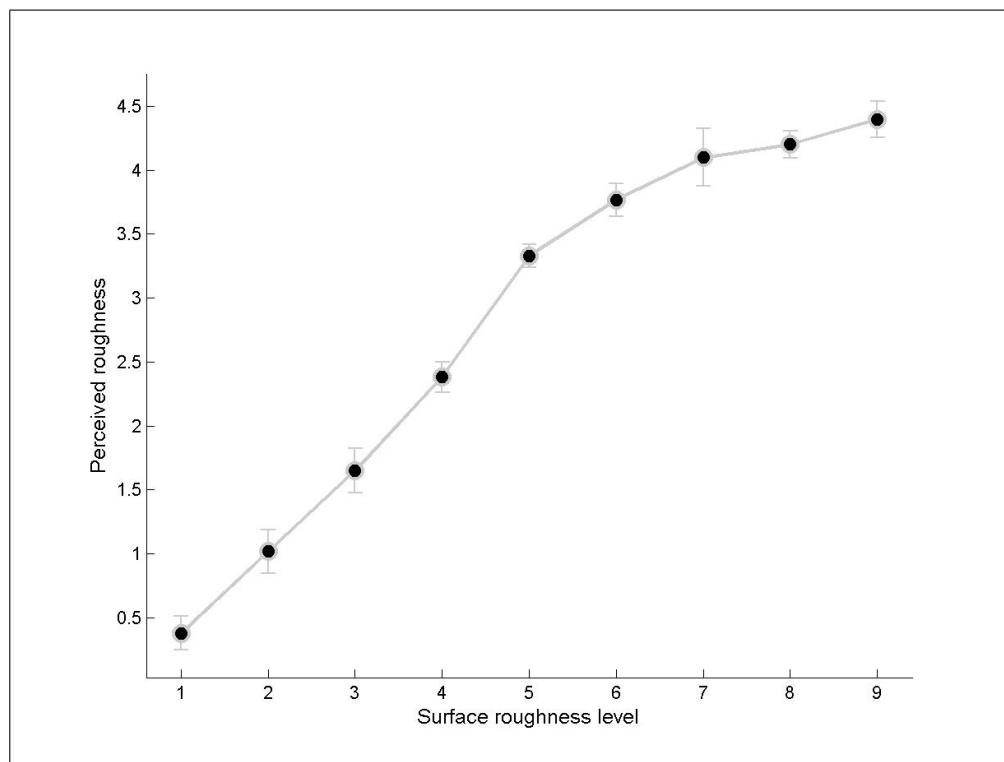
Una volta terminato l'esperimento, all'utente è stato presentato un semplice questionario contenente le seguenti domande, quasi tutte con risposta a scelta multipla:

1. Cosa ti è sembrato variasse tra i diversi scenari? [*La parte aptica | L'audio | Entrambi*]
2. Su cosa ti sei basato prevalentemente per giudicare la ruvidità? [*Sulla parte aptica | Sul suono | Su entrambi*]
3. Una superficie ruvida è caratterizzata da asperità; secondo te, nelle superfici che hai percepito come ruvide, la distanza tra le varie asperità: [*Rimane sempre costante per tutte le superfici ruvide | Varia da superficie a superficie ma resta costante sulla stessa superficie | Varia sempre, anche per la stessa superficie*]
4. L'aspetto grafico ha influenzato le tue valutazioni? [*Sì | No*]
5. Se sì, in che modo? A cosa ti ha fatto pensare? [*Risposta aperta*]
6. In base a quanto hai percepito, secondo te il cursore: [*Striscia sempre sulla superficie | Rotola sempre sulla superficie | In alcuni casi rotola, in altri striscia (in tal caso dire approssimativamente i casi in cui hai avvertito l'uno o l'altro tipo di movimento)*]
7. Secondo te, tra i vari scenari: [*Varia solo il materiale della superficie | Varia solo il materiale del cursore | Variano i materiali di entrambi | Non variano i materiali*]

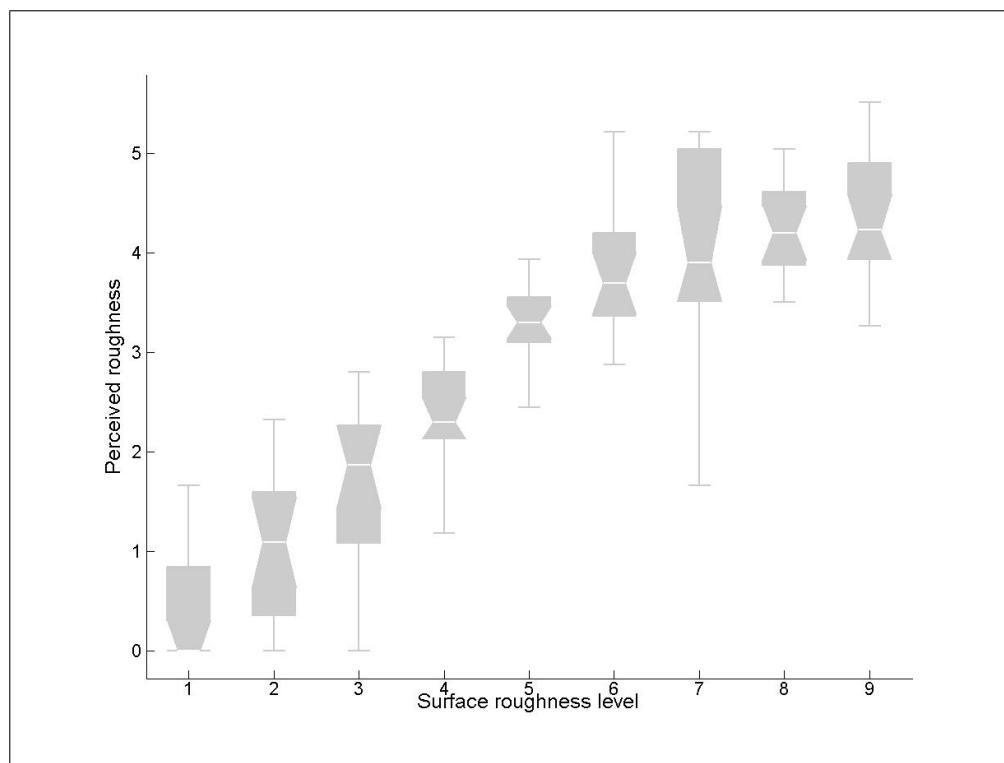
Tutti i soggetti hanno gentilmente offerto la loro partecipazione senza ricevere alcun compenso.

## 7.5 Esperimento 1

In questa prima serie di esperimenti vengono variati concordemente gli stimoli aptici e sonori. I partecipanti sono 20 soggetti, di età compresa tra i 21 e i 32 anni, 13 ragazzi e 7 ragazze. Quasi tutti i partecipanti hanno riportato delle stime concordi con i livelli di ruvidità simulati, come si può notare dal grafico in figura 7.3, il quale riporta la dipendenza tra ruvidità simulata (in ascissa) e la media della ruvidità percepita dai soggetti per lo stesso livello (in ordinata). Nei primi scenari (quelli utilizzati come training implicito), tutti i soggetti tendono a sovrastimare la ruvidità, giudicando molto ruvide anche le superfici più lisce. Tale problema viene però risolto dal training, il quale permette all'utente di capire



**Figura 7.3:** Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al primo esperimento.



**Figura 7.4:** Grafico boxplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al primo esperimento. La linea orizzontale rappresenta la mediana, le linee superiori e inferiori i limiti dei quartili, la linea verticale l'estensione dei dati.

bene la variazione della scala; negli scenari successivi infatti i soggetti iniziano ad utilizzare tutti i livelli di ruvidità a disposizione.

Analizzando il grafico di figura 7.3 si nota un andamento piuttosto lineare che spazia lungo tutti i valori possibili della scala data, la quale va da 0 a 5, dove 0 indica una superficie molto liscia e 5 una superficie molto ruvida. Il grafico in figura 7.4 dimostra invece che per alcuni scenari le stime variano considerevolmente; questa condizione è molto evidente ad esempio per il livello di ruvidità 7, e una spiegazione può essere ricercata analizzando la successione nella quale gli scenari vengono presentati all'utente. Gli scenari con livello di ruvidità 7 infatti vengono quasi sempre presentati subito dopo uno scenario di livello di ruvidità inferiore, ed è ipotizzabile quindi che sia questo ad influenzare la percezione. Se una superficie ruvida viene presentata subito dopo una superficie molto liscia, la prima viene stimata come più ruvida di quello che è in realtà; se invece vengono presentate successivamente due superfici poco diverse tra loro si otterranno stime simili per le due. Queste considerazioni trovano riscontro se si analizza la varianza dei risultati relativi al livello di ruvidità 5: gli scenari che simulano tale livello di ruvidità seguono sempre scenari che simulano ruvidità poco diverse, ed infatti le stime variano in un intervallo più limitato.

Due soggetti sono stati scartati dall'analisi in quanto considerabili come *outlier*, ovvero soggetti la cui media delle stime si discosta troppo dalla media di tutti gli altri partecipanti: entrambi non hanno saputo utilizzare pienamente la scala a disposizione.

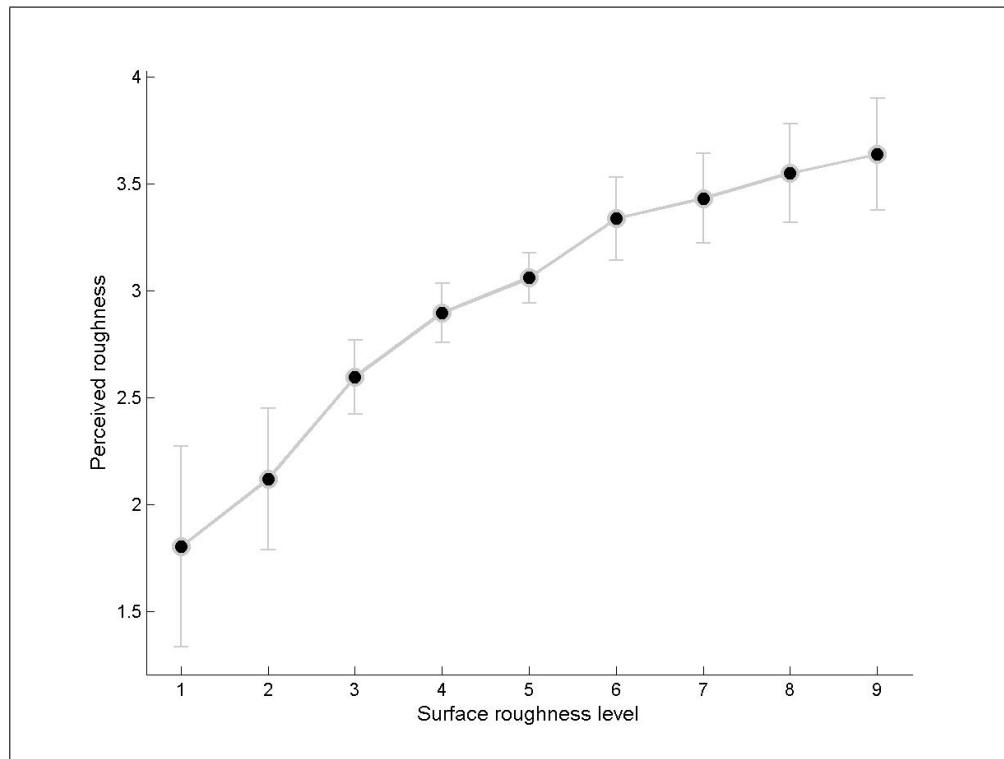
## 7.6 Esperimento 2

Nella seconda serie di esperimenti i partecipanti sono 20 soggetti di età compresa tra i 19 e i 30 anni, dei quali 14 ragazzi e 6 ragazze. Lo stimolo acustico viene variato (come descritto nel paragrafo 7.4) mentre quello aptico viene mantenuto costante ad un livello tale da simulare una superficie ruvida o poco ruvida ( $\beta = 0.8$ , nessuna amplificazione).

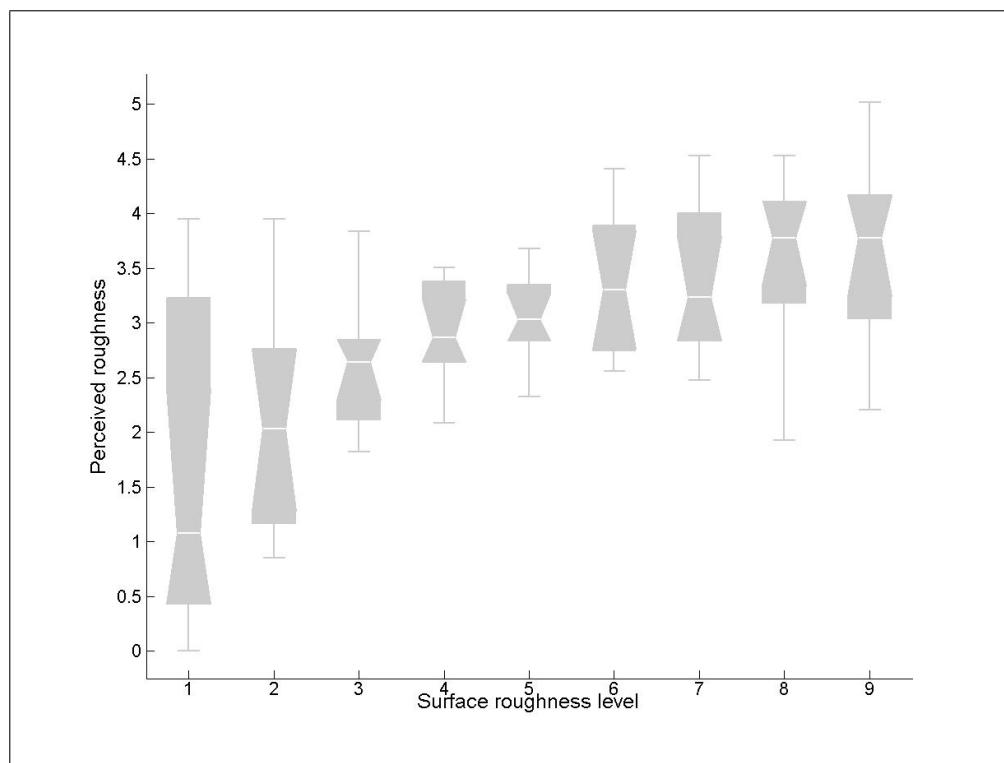
Anche in questo caso è stata confermata l'utilità del training implicito come strumento che permette all'utente di prendere confidenza con il dispositivo e con la scala imposta.

Dall'analisi delle stime è stato notato che la variazione discorde dei due stimoli ha creato molta confusione nei soggetti che si sono sottoposti a questa serie di esperimenti; infatti è risultato che:

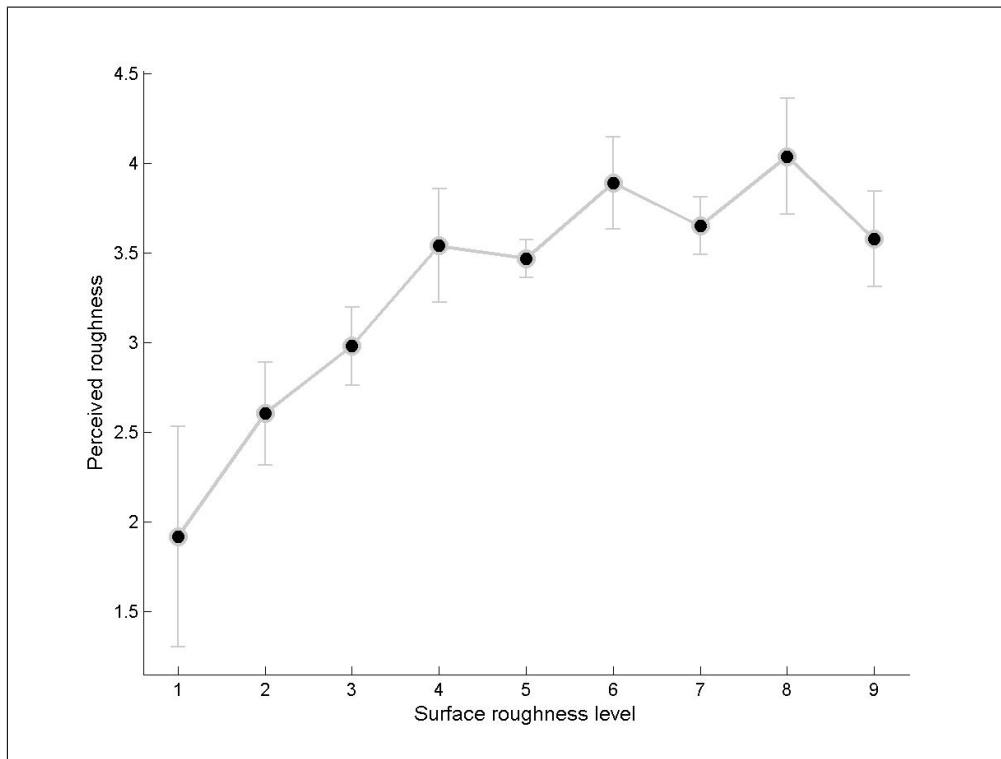
- più della metà degli utenti (11 soggetti) ha basato la propria stima o solo sullo stimolo acustico o su una combinazione di entrambi gli stimoli, produ-



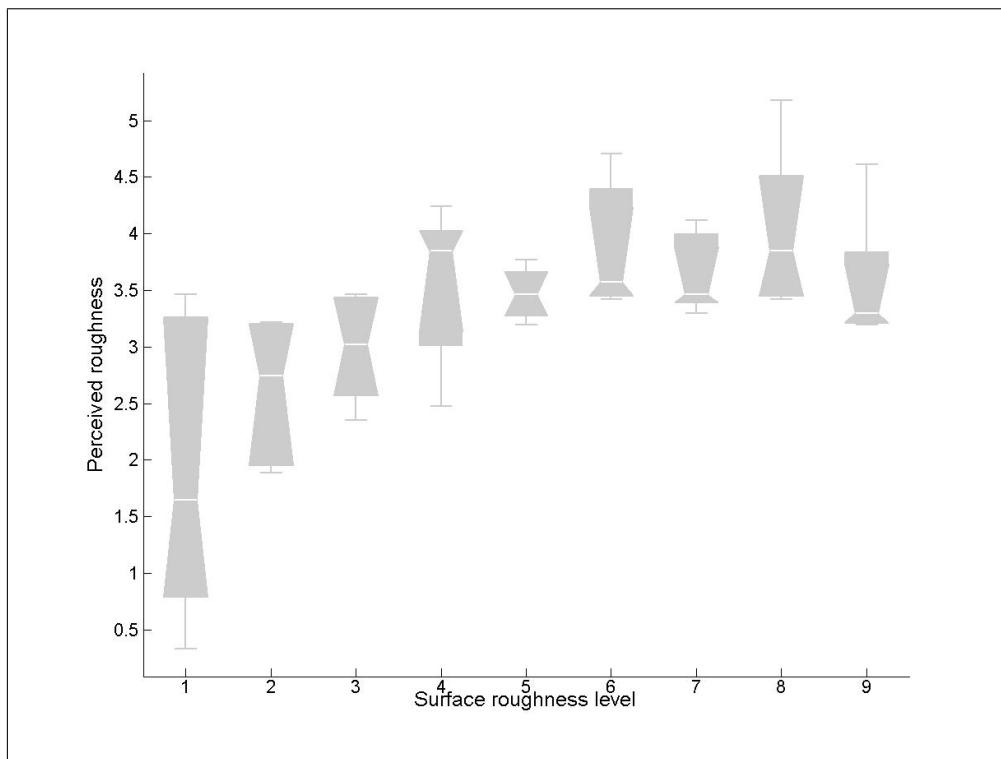
**Figura 7.5:** Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al secondo esperimento (primo gruppo).



**Figura 7.6:** Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al secondo esperimento (primo gruppo). La linea orizzontale rappresenta la mediana, le linee superiori e inferiori i limiti dei quartili, la linea verticale l'estensione dei dati.



**Figura 7.7:** Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al secondo esperimento (secondo gruppo).



**Figura 7.8:** Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al secondo esperimento (secondo gruppo). La linea orizzontale rappresenta la mediana, le linee superiori e inferiori i limiti dei quartili, la linea verticale l'estensione dei dati.

cendo risultati variabili e concordi con la variazione della stimolo acustico (tali soggetti verrano indicati come primo gruppo);

- alcuni soggetti non hanno preso in considerazione lo stimolo acustico o si sono fatti influenzare da questo in minima parte, producendo quindi stime della ruvidità che variano nel piccolo intervallo della scala compreso tra “poco ruvido” e “molto ruvido” (tali soggetti verrano indicati come secondo gruppo).

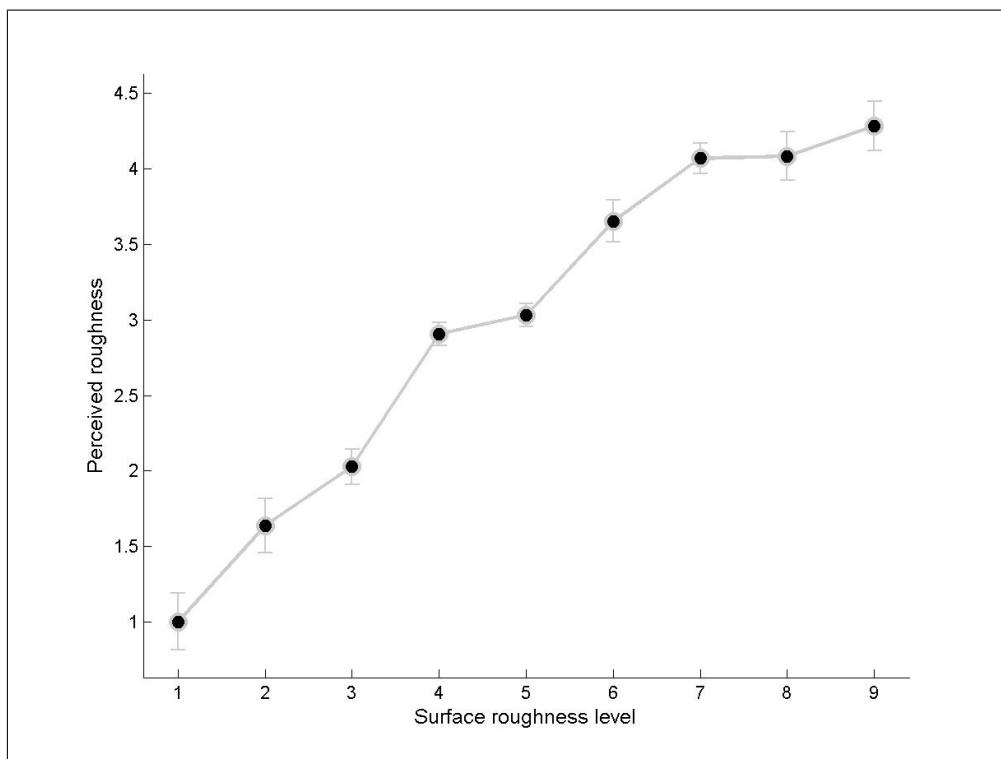
I grafici riportati in figura 7.5 e in figura 7.6 si riferiscono alle stime date dal primo gruppo. Nel primo dei due grafici è evidente un andamento lineare delle percezioni simile a quello ottenuto nel primo esperimento; tuttavia si nota come le variazioni più ampie si hanno in corrispondenza dei primi due scenari, ovvero quelli in cui la discordanza tra stimolo aptico e uditivo è maggiore (mentre l’audio simula una superficie liscia, il dispositivo aptico ne simula una ruvida). Inoltre, i quartili e le estensioni dei dati si allargano nelle due direzioni intorno al livello 5 (livello centrale), cioè aumentano le variazioni delle stime man mano che aumenta l’incoerenza tra gli stimoli: questo sembra supportare l’idea che la coerenza tra le modalità influenza la stima.

La stessa osservazione può essere fatta per il secondo gruppo. In questo caso però non sono state ottenute delle stime altrettanto lineari (figura 7.7 e figura 7.8), e la maggior parte di tali stime si colloca tra i livelli 3 e 4. L’andamento non lineare delle stime relative ai livelli dal quarto al nono può essere giustificato se si pensa che, basandosi solo sulla percezione aptica (che resta invariata), la scala delle risposte possibili diventa troppo ampia ed è difficile associare sempre lo stesso livello di ruvidità agli stessi scenari; l’alternanza infatti è sempre compresa tra i livelli 3 e 4, i quali corrispondono a superficie “poco ruvida” e “ruvida”.

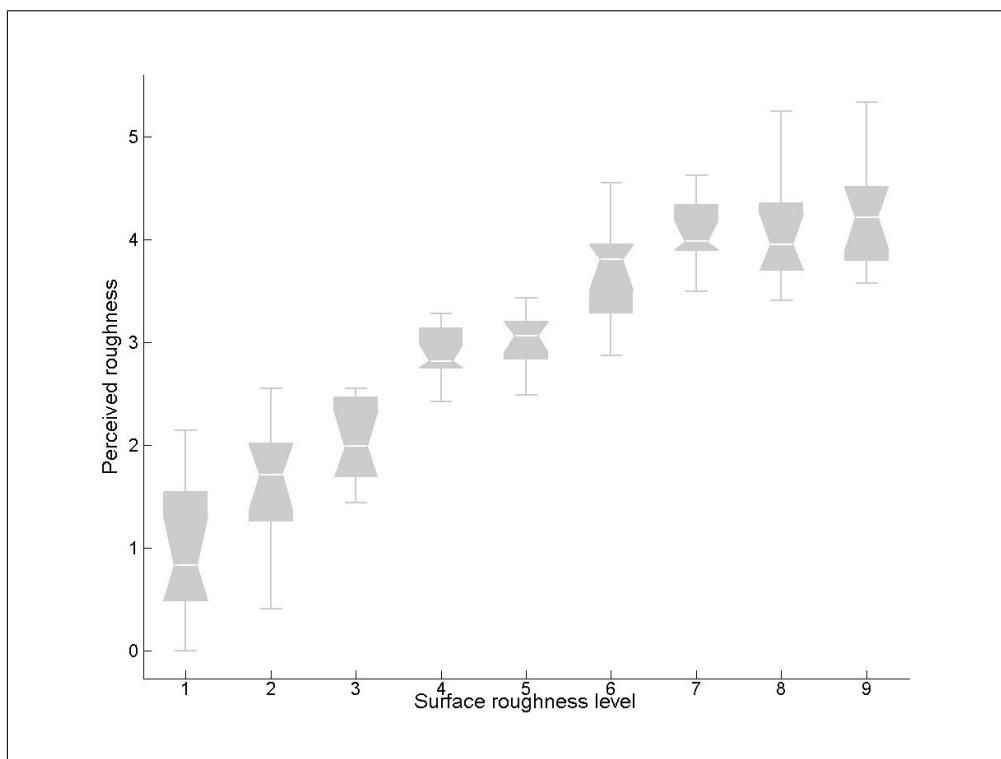
Infine 4 soggetti sono stati considerati outlier in quanto non hanno saputo utilizzare correttamente la scala a disposizione.

## 7.7 Esperimento 3

Alla terza serie di esperimenti hanno partecipato 14 soggetti, di età compresa tra i 20 e i 35 anni, 10 ragazzi e 4 ragazze. A differenza degli esperimenti precedenti, in questo caso viene variato solo lo stimolo aptico (come indicato nel paragrafo 7.4), mentre lo stimolo acustico resta costante ( $\beta = 0.8$ , nessuna amplificazione). Nuovamente il training隐式的 si è rivelato utile. Rispetto alla seconda serie di esperimenti, la variazione discorde degli stimoli ha indotto meno confusione nei partecipanti: alcuni di loro sono riusciti a discriminare correttamente le due variazioni, mentre molti hanno dichiarato di aver basato le loro



**Figura 7.9:** Grafico lineplot delle medie delle ruvidità percepite dai soggetti che hanno partecipato al terzo esperimento.



**Figura 7.10:** Grafico boxplot delle medie delle ruvidità percepite da alcuni tra i soggetti che hanno partecipato al terzo esperimento. La linea orizzontale rappresenta la mediana, le linee superiori e inferiori i limiti dei quartili, la linea verticale l'estensione dei dati.

percezioni maggiormente sullo stimolo aptico. Ciò si riflette nella linearità del grafico in figura 7.9. Anche qui è stato riscontrato un fenomeno simile a quello verificatosi nel secondo esperimento (anche se in misura minore): nei primi scenari (quelli dove la discordanza tra i due stimoli è massima) l'intervallo di variazione delle stime è molto ampio. In questo caso bisogna aggiungere che alcuni soggetti, ad esperimento finito, sostenevano di essere stati messi in difficoltà da una scala dei valori di ruvidità troppo ampia. Due soggetti non sono stati in grado di utilizzare i valori della scala e pertanto sono stati considerati outlier.

## 7.8 Risultati sulla confidenzialità delle risposte

Per esprimere il grado di sicurezza relativamente alle stime date, ogni soggetto poteva scegliere fra tre risposte: “molto sicuro”, “sicuro” e “poco sicuro”. Nella maggior parte dei casi i partecipanti hanno scelto l'opzione intermedia (cioè si sentivano “sicuri” della loro stima). Raramente invece si sono dichiarati “poco sicuri” (6 partecipanti al primo esperimento non hanno mai scelto questa risposta); per alcuni ciò è avvenuto in corrispondenza della stima di una superficie molto ruvida subito dopo una molto liscia, mentre qualcun altro invece si è dichiarato poco sicuro sulla percezione di una superficie di ruvidità poco diversa dalla precedente.

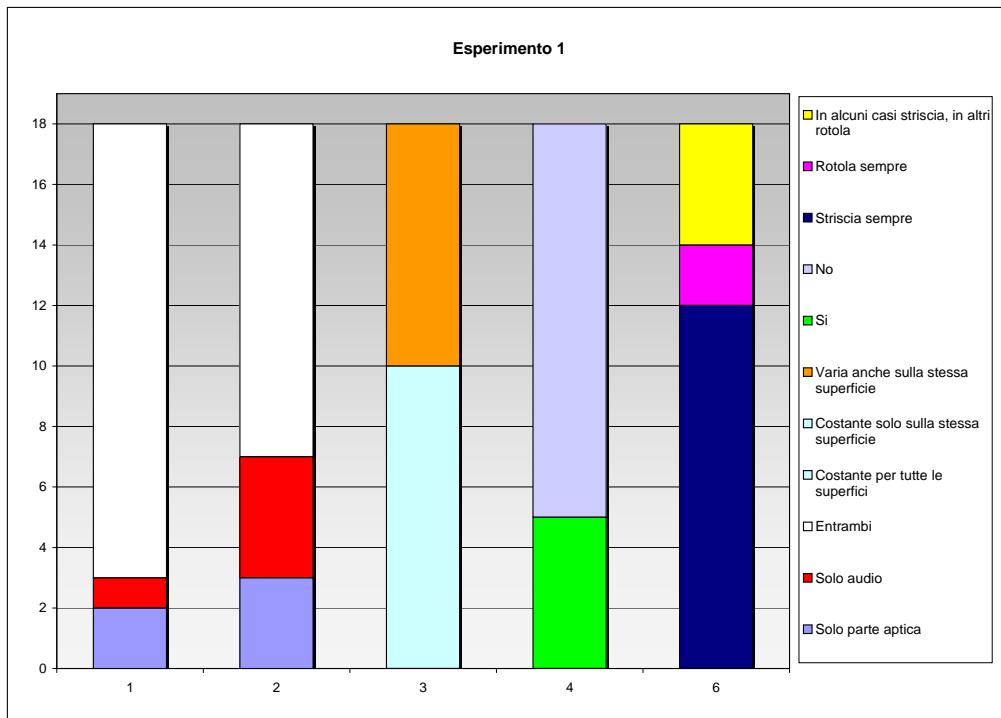
Nel secondo esperimento si è verificato un decremento della sicurezza, imputabile sicuramente alla variazione discorde degli stimoli. Il decremento della sicurezza nel terzo esperimento invece è stato molto più lieve.

Tutto ciò sembrerebbe confermare il fatto che le persone sono influenzate maggiormente dal senso del tatto rispetto a quello dell'udito. Tuttavia non è possibile effettuare una analisi esauriente di questo tipo basandosi sulla confidenzialità delle risposte, in quanto molti soggetti, in base a loro dichiarazioni al termine dell'esperimento, non hanno saputo utilizzare questo strumento.

## 7.9 Risultati sul questionario post-esperimento

### 7.9.1 Esperimento 1

La maggior parte dei partecipanti (15 su 18) hanno percepito una variazione di entrambi gli stimoli tra i diversi scenari, mentre 11 soggetti hanno dichiarato di aver basato le loro stime sia sullo stimolo aptico che sullo stimolo acustico. Tre soggetti, pur avendo percepito la variazione di entrambi gli stimoli, hanno basato la loro stima solo sulla percezione uditiva, dimostrando una scarsa confidenza nei confronti di ciò che hanno avvertito tramite il dispositivo aptico. Due soggetti



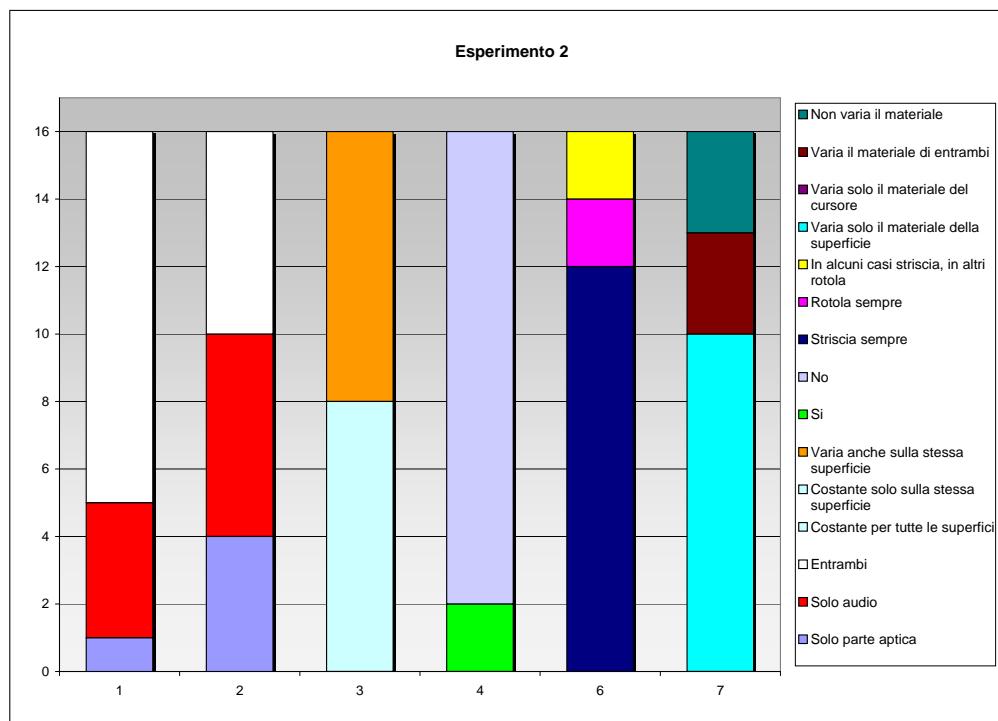
**Figura 7.11:** Risultati delle risposte al questionario post–perimentale relativo al primo esperimento; in ascissa è riportato il numero della domanda (in base all’ordine riportato nel paragrafo 7.4), in ordinata il numero di risposte.

hanno percepito solo una variazione della parte aptica, basando le loro stime solo su questa, mentre un solo soggetto ha avvertito una variazione solo dello stimolo acustico, e pertanto ha basato le sue stime solo su questo.

Riguardo alla distribuzione delle asperità sulla superficie, secondo il 50% dei partecipanti questa varia da superficie a superficie restando costante sulla stessa, mentre per il rimanente 50% si ha una variazione anche sulla stessa superficie. Dato che il profilo della superficie viene ricavato da un rumore frattale, la distribuzione delle asperità è del tutto casuale e varia anche sulla stessa superficie; tuttavia l’illusione di una distribuzione costante può essere dovuta al fatto che le asperità sono molto fitte.

Nonostante l’interfaccia grafica sia stata mantenuta la più semplice ed essenziale possibile, 5 soggetti hanno dichiarato di essere stati influenzati dall’aspetto grafico; tuttavia tale influenza dello stimolo visivo consiste semplicemente nel constatare la concordanza tra il movimento del dispositivo aptico e del cursore sullo schermo.

Per più della metà dei soggetti (12 su 18) il cursore strisciava sulla superficie, 2 sostengono che il cursore rotola, mentre secondo i restanti 4 tale sensazione varia: in corrispondenza di superfici più ruvide hanno avvertito un moto di sfregamento, mentre per le superfici lisce il cursore sembra rotolare.



**Figura 7.12:** Risultati delle risposte al questionario post-sperimentale relativo al secondo esperimento; in ascissa è riportato il numero della domanda (in base all'ordine riportato nel paragrafo 7.4), in ordinata il numero di risposte.

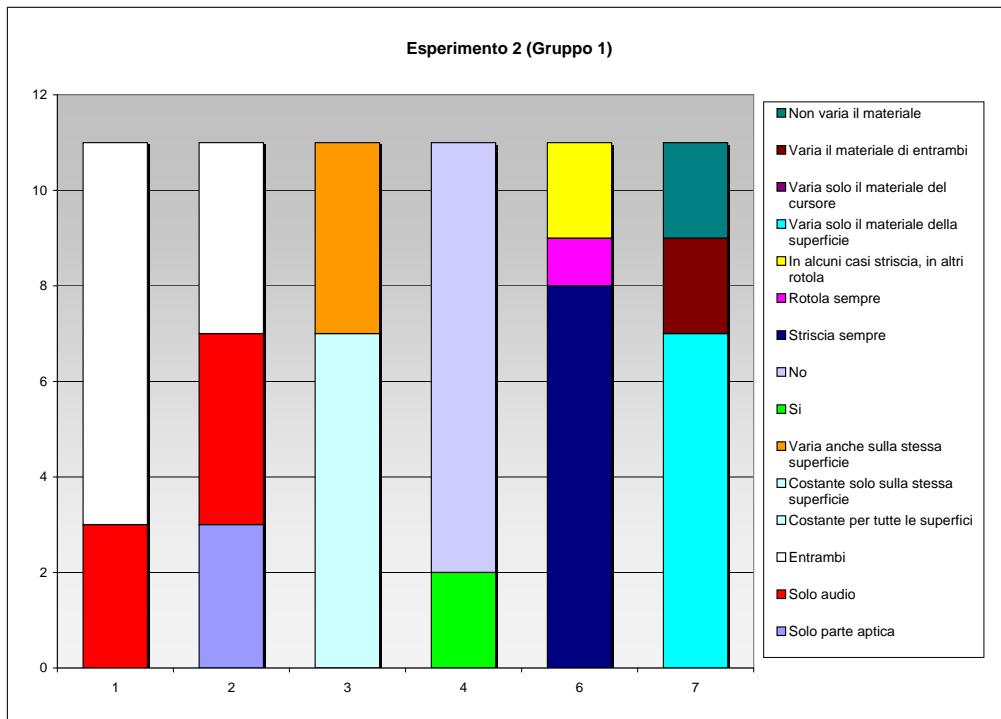
### 7.9.2 Esperimento 2

Pur avendo una variazione solo dello stimolo acustico, 11 soggetti su 16 hanno percepito una variazione di entrambi gli stimoli; tuttavia molti di loro hanno dichiarato di aver percepito una variazione minima della parte aptica in confronto alla parte acustica. I rimanenti partecipanti invece sostengono che varia solamente quest'ultima.

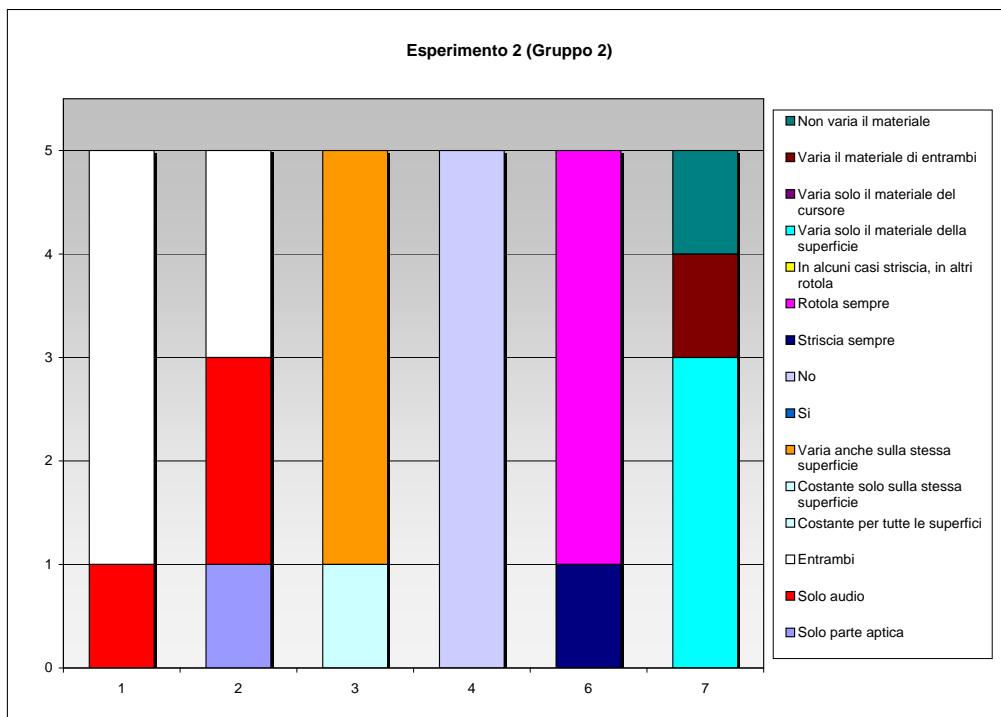
Due soggetti facenti parte del secondo gruppo hanno dichiarato di essersi basati prevalentemente sullo stimolo acustico per giudicare la ruvidità, anche se, analizzando le loro stime, si capisce come invece siano stati fortemente influenzati dallo stimolo aptico. Per contro, tre utenti del primo gruppo sostengono di aver basato le loro stime prevalentemente sullo stimolo aptico, mentre l'analisi delle stime dimostra che sono stati influenzati fortemente dall'audio. Sempre nel primo gruppo, i soggetti che hanno percepito una variazione di entrambi gli stimoli hanno basato le loro stime su entrambi, mentre quelli che hanno percepito solo una variazione dello stimolo acustico hanno basato le stime su questo.

Anche in questo esperimento le risposte relative alla distribuzione delle asperità si distribuiscono in modo uguale tra distribuzione regolare sulla stessa superficie e distribuzione variabile anche sulla stessa superficie.

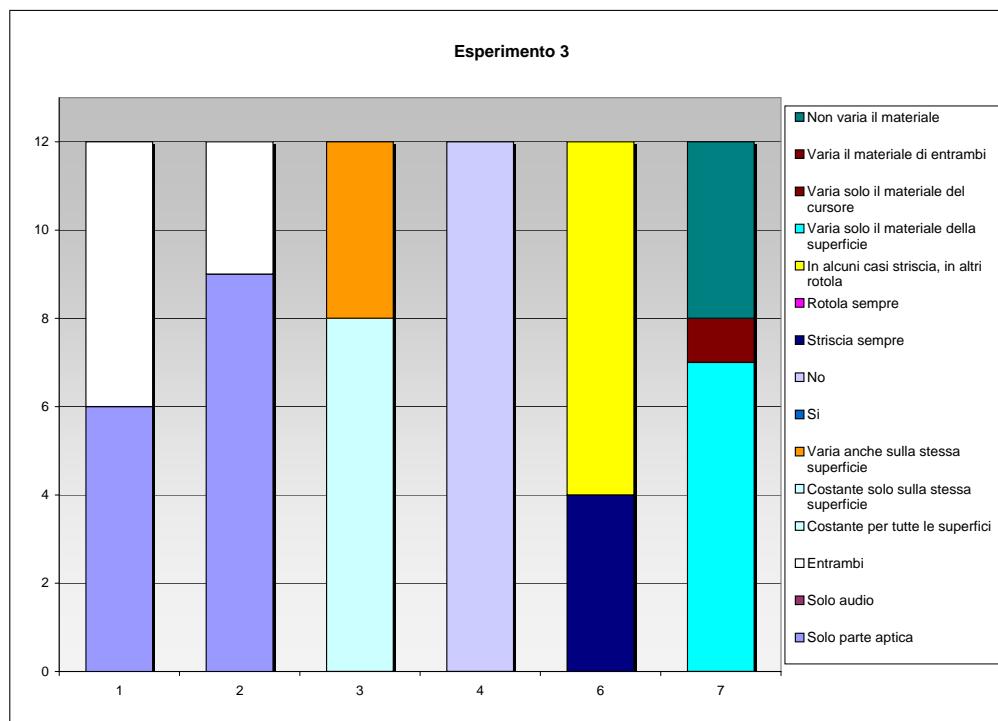
Solo due partecipanti dichiarano di essere stati influenzati dall'aspetto grafico:



**Figura 7.13:** Risultati delle risposte al questionario post-sperimentale relativo al secondo esperimento (primo gruppo); in ascissa è riportato il numero della domanda (in base all'ordine riportato nel paragrafo 7.4), in ordinata il numero di risposte.



**Figura 7.14:** Risultati delle risposte al questionario post-sperimentale relativo al secondo esperimento (secondo gruppo); in ascissa è riportato il numero della domanda (in base all'ordine riportato nel paragrafo 7.4), in ordinata il numero di risposte.



**Figura 7.15:** Risultati delle risposte al questionario post-sperimentale relativo al terzo esperimento; in ascissa è riportato il numero della domanda (in base all’ordine riportato nel paragrafo 7.4), in ordinata il numero di risposte.

per entrambi la visione di una superficie sempre liscia ha creato difficoltà nel capire come variava la ruvidità.

Secondo dodici soggetti il cursore striscia sempre sulla superficie; due hanno riportato invece che a volte il cursore sembra rotolare e altre volte strisciare (sulle superfici più lisce rotola mentre su quelle più ruvide striscia). Per altri due infine il cursore è sembrato rotolare sempre.

Rispetto al primo esperimento, è stato ritenuto opportuno chiedere ai partecipanti se hanno percepito una variazione dei materiali di cursore e superficie, in quanto superfici costituite di diverso materiale ma aventi livello di ruvidità simile producono suoni diversi. La maggioranza ha percepito una variazione del materiale della superficie; secondo tre soggetti invece variano i materiali di entrambi, mentre per altri tre i materiali non variano né per la superficie né per il cursore.

### 7.9.3 Esperimento 3

Secondo una metà dei partecipanti, tra i vari scenari cambia solo lo stimolo aptico, mentre l'altra metà ha avvertito una variazione di entrambi gli stimoli. Solo 3 soggetti hanno basato le loro stime su entrambi gli stimoli, tutti gli altri si sono basati esclusivamente sullo stimolo aptico; inoltre, i soggetti che nel questionario hanno dichiarato di aver basato le loro stime su entrambi gli stimoli hanno

poi specificato di essere stati influenzati in minima parte dallo stimolo acustico, basandosi principalmente su quello aptico.

Per 4 utenti le asperità sono distribuite in modo non costante anche sulla stessa superficie, mentre per tutti gli altri la distribuzione resta costante all'interno di una stessa superficie.

Nessuno è stato influenzato dall'aspetto grafico. I due terzi dei partecipanti hanno avvertito un moto di sfregamento tra cursore e superficie; i rimanenti, come è successo negli esperimenti precedenti, hanno dichiarato che sulle superfici più lisce il cursore sembrava rotolare, mentre in presenza di superfici ruvide hanno avvertito uno sfregamento.

Secondo la maggior parte dei soggetti (7 su 12) tra i vari scenari viene variato il materiale della superficie, mentre quello del cursore resta costante. Un solo soggetto ha avvertito una modifica del materiale del cursore, mentre per tutti gli altri i materiali non cambiavano mai.

## 7.10 Conclusioni sperimentali

Negli studi condotti è stata confermata la predominanza del senso del tatto sull'udito, anche nel caso in cui l'utente non sia chiamato ad esplorare oggetti reali con l'uso delle mano ma oggetti virtuali con l'uso di un dispositivo aptico. Nonostante questa predominanza, si è visto come lo stimolo acustico in molti casi modula quello aptico: infatti, più della metà dei partecipanti ha avvertito una variazione di entrambi gli stimoli quando invece solo uno dei due veniva variato tra i diversi scenari.

La scala dei valori di ruvidità utilizzata si è rivelata troppo ampia per molti soggetti: questi si sono trovati in difficoltà a scegliere tra i valori "molto liscia" e "liscia" e tra "poco ruvida" e "ruvida". Questa difficoltà è stata anche riscontrata nei partecipanti classificati come outlier; pur non avendo questi soggetti riportato osservazioni in proposito, analizzando le loro stime più nel dettaglio si nota una confusione nell'utilizzo dei valori estremi della scala. Non sappiamo però se una scala ristretta a quattro valori (invece dei sei attuali) possa risultare corretta o troppo restrittiva.

L'indagine sulla confidenzialità delle risposte non è stata condotta nel modo ottimale. Come abbiamo visto, molti utenti non hanno prestato molta attenzione a tale domanda, ritenendola poco importante o, in alcuni casi, addirittura noiosa. Probabilmente poteva essere sufficiente formulare solamente un quesito sul tema nel questionario post-sperimentale.

Al di là di questi accorgimenti, come è emerso soprattutto dai risultati del primo esperimento, è stato raggiunto l'obiettivo di creare un'applicazione di realtà

virtuale per l’interazione continua e bimodale con superfici simulate. La componente aptica influenza molto la percezione dell’utente, mentre la componente sonora, oltre ad aumentare il realismo degli scenari virtuali, fornisce informazioni utili quando i limiti del dispositivo aptico non permettono un’analisi accurata della tessitura caratterizzante la superficie esplorata.



# Conclusione

In questa tesi è stata illustrata la progettazione e la realizzazione di un'applicazione per l'interazione audio–aptica con un ambiente virtuale, orientata alla percezione bimodale delle tessiture di una superficie. Sono stati descritti gli strumenti hardware e software utilizzati e come sono state implementate separatamente la simulazione aptica e la simulazione sonora. Successivamente è stato descritto come le due componenti sono state integrate tra loro, utilizzando l'applicazione risultante per condurre alcuni esperimenti sulla percezione audio–aptica.

Gli algoritmi di sintesi audio hanno prodotto buoni risultati, riuscendo a trasmettere la sensazione di ruvidità di una superficie durante l'esplorazione di questa. Lo stesso può essere detto relativamente all'implementazione dell'interfaccia aptica, anche se l'utilizzo di un dispositivo con una capacità di risoluzione più elevata potrebbe permettere una simulazione più dettagliata e realistica delle tessiture, permettendo all'utente di cogliere la presenza di tutte le asperità collocate sulla superficie quasi come se usasse le sue mani nell'esplorazione. La latenza tra gli stimoli audio e aptico è ridotta al minimo, cosicché nessuno stimolo viene percepito come estraneo all'esplorazione.

Risulta invece migliorabile l'interfaccia grafica, in particolare per quanto riguarda la gestione di luci ed ombre. Queste infatti sono componenti essenziali per il realismo dello scenario e per permettere all'utente una migliore interazione data da una corretta percezione delle posizioni degli oggetti virtuali l'uno rispetto all'altro. Tutto questo però si rende necessario solo se vengono realizzati scenari più complessi; abbiamo visto come in uno scenario composto da un oggetto e un piano le semplici tecniche utilizzate sono risultate sufficienti. Infine sono da modificare alcune scelte nella realizzazione degli esperimenti, quali la scala disponibile per le stime di ruvidità e la modalità di raccolta dei dati sulla confidenzialità delle risposte.

Il codice è dipendente dalla piattaforma solo nelle funzioni relative alla scrittura e lettura in memoria dei dati; la parte rimanente, le librerie utilizzate e l'ambiente di sintesi audio sono multipiattaforma, rendendo molto semplice il porting su altri sistemi operativi. Sebbene l'implementazione qui descritta sia stata orientata alla realizzazione di un sistema sperimentale, i campi di utilizzo

---

di questa applicazione sono moltissimi. Grazie alla struttura modulare, è possibile l'implementazione di diversi algoritmi fisici di sintesi sonora, la modifica dell'interfaccia grafica per rappresentare scenari sempre più completi o creare altri tipi di effetti aptici.

L'applicazione realizzata può essere usata quindi per altre tipologie di esperimenti, o come base per la creazione di ambienti virtuali più complessi utilizzabili nell'intrattenimento (interazione videoludica), nell'addestramento medico (operazioni chirurgiche), per simulazioni militari (controllo dei robot per il disinnescaggio di ordigni esplosivi) e per creare strumenti di aiuto alle persone diversamente abili (nuove interfacce per facilitare l'utilizzo di strumenti informatici).

# Bibliografia

- [1] Avanzini, F., Rath, M., e Rocchesso, D. Low-level sound models: resonators, interactions, surface textures. *The Sounding Object*, pp. 119–148, 2003.
- [2] Barbagli, F., Salisbury, K., e Devengenzo, R. Enabling multifinger, multi-hand virtualized grasping. *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA 03)*, vol. 1, pp. 806–815, 2003.
- [3] Basdogan, C., Ho, C. H., e Srinivasan, M. A. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. *Proc. ASME Dynamic Systems and Control Division*, 61:77–84, 1997.
- [4] Chapentier, A. *Experimental study of some aspects of weight perception*, volume 3. Archives de Physiologie Normales et Pathologiques, 1891.
- [5] Corsini, G. e Saletti, R. A  $1/f^\gamma$  power spectrum noise sequence generator. *IEEE Trans. on Instrumentation and Measurement*, 37(4):615–619, Dicembre 1988.
- [6] Crosato, P. Una piattaforma per il rendering audio–aptico di interazioni di contatto, 2006. Tesi di Laurea, Università degli studi di Padova.
- [7] DiFilippo, D. e Pai, D. K. The AHI: an audio and haptic interface for contact interactions. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pp. 149–158. ACM Press, 2000.
- [8] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., e Worley, S. Texturing and modeling: a procedural approach. *AP Professional*, 1994.
- [9] Ernst, M. O. e Bülthoff, H. H. Merging the senses into a robust percept. *Trends in Cognitive Sciences*, 8(4), Aprile 2004.
- [10] Fritz, J. P. e Barner, K. E. Stochastic models for haptic texture. *Proceedings of SPIE's International Symposium on Intelligent System and Advanced Manufacturing - Telemanipulator and Telepresence Technologies III*, 1996.

## BIBLIOGRAFIA

---

- [11] Gaver, W. W. How do we hear in the world? Explorations of ecological acoustics. *Ecological Psychology*, 5(4):285–313, 1993.
- [12] Gaver, W. W. What in the world do we hear? An ecological approach to auditory event perception. *Ecological Psychology*, 5(1):1–29, 1993.
- [13] Gibson, J. J. Observation on active touch. *Psychological Review*, 69:477–490, 1962.
- [14] Grill, T. *flext An Introduction*. Institut für Komposition und Elektroakustik. <http://www.parasitaere-kapazitaeten.net/ext/flext/>.
- [15] Guest, S., Catmur, C., Lloyd, D., e Spence, C. Audiotactile interactions in roughness perception. *Experimental Brain Research*, 146(2):161–171, July 2002.
- [16] Haruyama, S. e Barsky, B. A. Using stochastic modeling for texture generation. *IEEE Computer Graphics and Applications*, pp. 7–19, 1984.
- [17] Hastings, H. M. e Sugihara, G. *Fractals: A User's Guide for the Natural Sciences*. Oxford University Press, 1993.
- [18] Kilgard, M. J. *The OpenGL Utility Toolkit (GLUT) Programming Interface*. Silicon Graphics, Inc., 1996. <http://www.opengl.org/resources/libraries/glut.html>.
- [19] Klatzki, R. L. e Lederman, S. J. Touch In *Experimental Psychology*. A cura di Healy, A. F. e Proctor, R. W., capitolo 6, pp. 144–176. Psychology Press, 2002.
- [20] Lederman, J. S. e Klatzky, R. L. Multisensory texture perception In *The Handbook Of Multisensory Processes*. A cura di Calvert, G. A., Spence, C., e Stein, B. E., pp. 107–122. MIT Press, 2004.
- [21] Lederman, S. J. Auditory texture perception. *Perception*, 8(1):93–103, 1979.
- [22] Lederman, S. J., Klatzki, R. L., Hamilton, C., e Morgan, T. Integrating multimodal information about surface texture via a probe: Relative contributions of haptic and touch produced sound sources. In *10th Annual meeting of Haptic Interfaces for Teleoperator and Virtual Environment Systems*. IEEE, 2002.
- [23] Lederman, S. J. e Klatzky, R. L. Hand movements: A window into haptic object recognition. *Cognitive Psychology*, 19:342–368, 1987.

- [24] Lederman, S. J. e Klatzky, R. L. The haptic glance: A route to rapid object identification and manipulation. *Attention and Performance*, 17: Cognitive regulation of performance: Interaction of theory and application:165–196, 1999.
- [25] Lederman, S. J., Klatzky, R. L., Collins, A., e Wardell, J. Exploring environments by hand or foot: Time-based heuristics for encoding distance in movement space. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13:606–614, 1987.
- [26] Lederman, S. J., Thorne, G., e Jones, B. Perception of texture by vision and touch: Multidimensionality and intersensory integration. *Journal of Experimental Psychology: Human Perception and Performance*, 12(2):169–180, 1986.
- [27] Levitin, D. J., Mathews, M. V., e MacLean, K. The perception of cross-modal simultaneity. In *Proc. of International Journal of Computing Anticipatory Systems*, 1999.
- [28] Loomis, J. M. e Lederman, S. J. Tactual perception. *Handbook of perception and human performances*, 2:31–41, 1986.
- [29] Luciani , A. Interaction as exchanged actions and their role in visual and auditory feedbacks. <http://www.interdisciplines.org/enaction/papers/14/version/original>.
- [30] Max, N. L. e Becker, B. G. Bump shading for volume texture. *IEEE Computer Graphics and Applications*, (14):18–20, 1994.
- [31] McNeely, W., Puterbaugh, K., e Troy, J. Six degree-of-freedom haptic rendering using voxel sampling. *Proc. ACM Siggraph*, pp. 401–408, 1999.
- [32] Millar, S. Memory in touch. *Psicothema*, 11:747–767, 1999.
- [33] Mitra, S. K. *Digital Signal Processing: A Computer Based Approach*. McGraw-Hill, 1998.
- [34] Otaduy, M.A. e Lin, M. Sensation preserving simplification for haptic rendering. *Proc. ACM Siggraph*, pp. 543–553, 2003.
- [35] Pentland, A. P. Fractal-based description of surfaces. *Natural Computation*, pp. 279–298, 1988.
- [36] Specifications for the PHANTOM® Omni™ haptic device. <http://www.sensable.com>.

## BIBLIOGRAFIA

---

- [37] Puckette, M. S. *Pd Documentation*. [http://crca.ucsd.edu/~msp/Pd\\_documentation/](http://crca.ucsd.edu/~msp/Pd_documentation/).
- [38] Resnick, S. *Adventures in Stochastic Processes*. Birkhäuser Boston, 1992.
- [39] Saletti, R. A comparison between two methods to generate  $1/f^\gamma$  noise. *Proc. IEEE*, 74:1595–1596, Novembre 1986.
- [40] Sensable Technologies. *OpenHaptics Toolkit API reference*, 2005. <http://www.sensable.com>.
- [41] Sensable Technologies. *OpenHaptics Toolkit Programmer's guide*, 2005. <http://www.sensable.com>.
- [42] Siira, J. e Pai, D. K. Haptic texturing - a stochastic approach. *International Conference on Robotics and Automation*, 1996.
- [43] Srinivasan, M. A. e Basdogan, C. Haptics in virtual environments: Taxonomy, research status, and challenges. *Comput & Graphics*, 21(4):393–404, 1997.
- [44] Srinivasan, M. A. e Morgenbesser, H. B. Force shading for haptic shape perception. *Proceedings of the ASME Dynamic System and Control Division*, 58:407–412, 1996.
- [45] Stone, R. J. Haptic feedback: A potted history, from telepresence to virtual reality In *Haptic Human-Computer Interaction: First International Workshop, Glasgow, UK*. A cura di Springer Berlin. Settembre 2000.
- [46] Van den Doel, K., Kry, P. G., e Pai, D. K. FoleyAutomatic: Physically-based sound effects for interactive simulation and animation. In *Computer Graphics (ACM SIGGRAPH 2001 Conference Proceedings)*, Agosto 2001.
- [47] Wornell, G. W. *The Digital Signal Processing Handbook*. CRC Press and IEEE Press, 1998.
- [48] Wright, R. S. Jr. e Lipchak, B. *OpenGL® SuperBible, Third Edition*. Sams Publishing, 2004.
- [49] Zilles, C. B. e Salisbury, J. K. A constraint based god-object method for haptic display. *IEEE International conference on intelligent robots and systems*, 1995.
- [50] Zmölnig, J. M. *HOWTO write an external for puredata*. Institut for electronic music and acoustics. <http://iem.kug.ac.at/pd/externals-HOWTO/>.