UNIVERSITÀ DI PADOVA **FACOLTÀ DI INGEGNERIA**

TESI DI LAUREA

Kinect strumento di localizzazione in uno spazio controllato

Laureando: Matteo Casagrande

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

Corso di Laurea Triennale in Ingegneria dell'Automazione

29 novembre 2011

Anno Accademico 2010/2011

Prefazione

Lo sviluppo continuo di applicazioni di realtà virtuale e realtà aumentata richiede strumenti dotati di una tecnologia sempre più avanzata e costi sempre più contenuti.

Il progetto esposto ha l'obiettivo di creare una delle tante possibili applicazioni per PC che sfrutta il dispositivo Kinect, introdotto in commercio alla fine del 2010 dalla Microsoft. Tale dispositivo è nato principalmente come controller per videogames, per la consolle XBOX 360, ma le capacità hardware che ha in dotazione e il prezzo contenuto grazie ad una economia di scala hanno attirato l'attenzione di numerosi sviluppatori software e ricercatori.
L'applicazione chiamata *Localization in Controlled Space* (LCS), argomento centrale di questa tesi, oltre a verificare le potenzialità di Kinect, si propone come strumento utile ai fini della realizzazione di ambienti audio 3D per applicazioni *mixed reality*.
LCS utilizza il dipositivo Kinect per tracciare un punto in corrispondenza della testa di una persona posta all'interno di una stanza virtuale controllata, allo stesso tempo crea e da' la possibilità di modificare la posizione di una sorgente sonora puntiforme all'interno di tale spazio. L' operazione di *head-tracking* eseguita da LCS verifica il ruolo che il software può assumere per lo sviluppo di applicazioni binaurali in ambienti audio 3D per la realtà aumentata.

Al termine del lavoro di sviluppo, sono presentate una breve analisi delle problematiche riscontrate durante il progetto e una valutazione delle proposte di possibili miglioramenti nell'utilizzo del dispositivo: sono valutati i suoi limiti, le possibilità di combinare Kinect a differenti tecnologie esistenti in commercio.

Sommario

Utilizzando lo strumento Microsoft Kinect, si vuole realizzare un'applicazione di realtà aumentata nella quale la sorgente sonora virtuale viene renderizzata intorno ad un ascoltatore immerso nello spazio controllato da Kinect. Grazie a tecnologie di audio binaurale, la sorgente sonora viene mossa intorno all'ascoltatore e i suoi movimenti, catturati dal dispositivo, hanno lo scopo di rendere la simulazione più immersiva possibile.

Primo capitolo: espone gli ambienti *mixed reality*, composti dalla realtà aumentata e dalla realtà virtuale. Si focalizza in primo luogo sulla sintesi dei gesti e sulla loro analisi, la quale comprende le tecniche di cattura, estrazione e mappatura delle caratteristiche e descrive i campi nei quali vengono applicate tali tecniche. In secondo luogo viene sottolineata l'importanza dell'*head-tracking* per applicazioni di renderizzazione audio o applicazioni di audio binaurale.

Secondo capitolo: introduce Microsoft Kinect specificando come nasce, i campi di utilizzo, l'impatto commerciale e fornisce una scheda tecnica delle caratteristiche del dispositivo. Offre inoltre una panoramica per l'utilizzo *cross-platform*, specificando le principali differenze legate alle varie tipologie di installazione.

Terzo capitolo: è il cuore dell'elaborato. Descrive l'intero progetto di programmazione partendo dal metodo sperimentale adottato per la raccolta dati e il dimensionamento della stanza virtuale controllata. Mostra come vengono calcolate le funzioni di cambiamento di coordinate relative alla posizione della testa all'interno della stanza virtuale. Infine giustifica le modifiche dell'interfaccia grafica apportate ad un'applicazione di partenza, specificando come vengono disegnati i due punti, sorgente sonora e *head-point*.

Nella parte conclusiva, racchiusa all'interno del capitolo 4, è riassunto tutto il lavoro svolto, sono esposte le problematiche emerse durante lo sviluppo del progetto ed alcuni possibili utilizzi dell'applicazione. Sono, infine, discussi i limiti del dispositivo Kinect e proposti dei miglioramenti per superare i limiti legati alle prestazioni dello strumento.

Ringraziamenti

Innanzi tutto desidero ringraziare il mio relatore Prof. Federico Avanzini per la continua disponibilità e la cordialità dimostratami durante il progetto e l'Ing. Michele Geronazzo per avermi aiutato a realizzare il mio primo progetto di programmazione, per aver saputo rispondere con pazienza alle mie domande e per avermi seguito in prima persona durante la stesura della tesi.

La fine degli studi segna sicuramente la fine di un enorme capitolo della propria vita e la mia tesi di laurea arriva in corrispondenza di un periodo della mia vita molto particolare. Nonstante i bei ricordi che mi legano alla vita universitaria, nonostante tutte le persone che ho avuto modo di conoscere in questi anni, ricorderò con molta gioia questi momenti ma non rimpiango assolutamente nulla di tutto ciò che sto per lasciare. Credo che l'idea di una vita stabile possa finalmente aiutarmi a fare quel passo decisivo per crescere in modo più equilibrato e sereno. Chi mi conosce lo sa, sa che questa laurea, anche se triennale, è comunque un risultato personale importantissimo e sudato. Lo studio si sa, non è mai stato il mio punto forte, ma grazie alla determinazione, l'impegno (anche se non sempre costante) e al sostegno di molti amici, parenti e persone care, sono riuscito a tagliare questo traguardo.
Ho scoperto che prima di studiare è fondamentale imparare a studiare, cosa che raggiunta la maturità ancora non sapevo fare. Amaramente pentito del mio negligente passato ho dovuto spendere i primi due anni della mia lunga vita universitaria per ambientarmi e prendere il giusto passo per poter proseguire fino alla fine. Naturalmente il risultato ripaga ampiamente lo sforzo e allevia gli errori commessi in passato. Non posso non pensare in questo momento anche a tutti coloro che sono stati miei professori sia alle scuole medie che alle scuole medie superiori, in particolare ai professori che mi hanno sempre mal giudicato e che non avrebbero scommesso un centesimo sulla mia laurea. Con una punta di sarcasmo dedico una piccola fettina della mia festa di laurea anche a loro.

È complicato partire dal principio e ringraziare tutti uno per uno poichè le persone che nel bene o nel male mi hanno accompagnato per tutta questa strada sono davvero tante e la strada è stata davvero lunga.
Ringrazio di cuore i miei primi coinquilini con i quali ho condiviso l'appartamento. Auguro davvero a tutti gli studenti che avranno intenzione di andare a vivere lontani da casa di trovare delle persone cosí oneste, educate e rispettose con cui condividere ogni singolo giorno sotto lo stesso tetto. Ringrazio particolarmente Giacomo (l'artista) per l'amicizia che ci lega nonostante le nostre strade da studenti si siano separate diverso tempo fa.
Come non citare le mie favolose e bellissime amiche Clara e Marta che tanto hanno provato, a volte con molto successo, a distogliermi dagli impegni scolastici facendomi apparire meno serio di quello che sono! Da ottime amiche si sono pure trasformate in ottime coinquiline regalandomi una convivenza vivace, rispettosa e felice. Grazie di cuore.
Ai miei amici della biblioteca di Marostica, che hanno dato un tocco di allegria durante le lunghe e noiose giornate di studio, talmente tanta allegria da sentire il dovere di ritrovarsi in biblioteca anche senza alcun esame in vista. è praticamente nata una seconda compagnia.

Oltre a ringraziare, mando un forte abbraccio ed un grosso bacio a Laura. Una ragazza d'oro che mi ha sempre apprezzato per quello che sono. Una ragazza che nel bene o nel male è sempre stata al centro della mia vita sentimentale, parecchio travagliata. Mi ha sempre sostenuto sia nella vita che nello studio e di certo merita molto ma molto di più che un ringraziamento. Un bacio grande.

Naturalmente non manca che ringraziare tutta la mia famiglia. Non essendo possibile citare tutta la numerosa parentela, un ringraziamento preventivo va a tutti gli zii, cugini e parenti vari che sempre si sono sempre interessati al mio studio.
Ringrazio di cuore mio zio Gianmichele, per il forte sostegno che mi ha sempre dato (ho sostenuto e superato un esame che volevo saltare a causa della scarsa preparazione solo perchè mio zio mi aveva fatto promettere almeno di provarci).
Mia nonna Maria che ad ogni esame mi dedicava una preghiera, mio nonno Antonio (Tony) che mai ha perso occasione di chiedermi come procedessero i miei studi e di ascoltare la descrizione di argomenti d'esame che a lui parevano interessanti.
Grazie di cuore a mia nonna Elda, non ha mai dubitato un solo istante che l'università fosse un traguardo troppo importante per me e mi ha sempre aiutato economicamente (sembrerà frivolo ma è pur sempre una cosa importante per uno studente).

A mio fratello più che un ringraziamento vorrei augurargli che possa arrivare presto al diploma di laurea per godersi il risultato, vorrei inoltre rassicurarlo con la mia esperienza di vita universitaria dadogli ogni qual volta ne abbia bisogno, consigli e motivazioni per affrontare questo difficile percorso.
Ringrazio di cuore mio papà che mi ha praticamente obbligato almeno a provare ad andare avanti negli studi. Senza questa forte spinta di sicuro avrei abbandonato o rinunciato alla vita universitaria e a tutti i bei ricordi e alla formazione intellettuale che mi ha lasciato, lo ringrazio anche per avermi dato l'opportunità di potermi ambientare gradualmente nel mondo del lavoro, assegnandomi compiti sempre più impegnativi curando cosí anche la mia formazione professionale.
Un forte e sentito abbraccio a mia mamma. Inevitabilmente si tende a scivolare nel banale quando si parla della propria mamma, ma in fin dei conti lei c'è sempre per qualsiasi cosa, sopporta i miei sbalzi d'umore, si prende sempre cura di me e di tutta la famiglia. Il suo lavoro è di certo quello più in ombra, ma è indubbiamente il più importante di tutti. Sicuramente la persona che più mi ha supportato, incoraggiato ed ha sempre dimostrato una ficucia incondizionata in me. Grazie mamma.

Un particolare pensiero va a mio zio Leopoldo. A lui devo davvero molto, moltissimo, sotto tutti i punti di vista. Che possa riposare in pace e godersi altrove la mia proclamazione di laurea. Ciao zio, tuo nipote si laurea!

Matteo

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Realtà aumentata e realtà virtuale

Da anni il continuo studio sulla realtà virtuale è oggetto di ricerca per un vasto numero di studiosi, scienziati e tecnici informatici. Tale ricerca racchiude in sè una gamma vastissima di applicazioni basate sulla realtè virtuale, a partire dai videogiochi, fino ad arrivare ad applicazioni nei campi della medicina e psicologia. Il termine *virtual reality* (VR), come afferma Benjamin Woolley nel suo libro "*Mondi Virtuali*" [5] fu coniato da Jaron Lanier nel 1989 durante il convegno SIGGRAPH, uno dei pionieri in questo campo che ha fondato la compagnia VPL Research (*Virtual Programming Languages*, linguaggi di programmazione virtuale).

Altre ricerche sono invece dedicate allo studio sulla *augmented reality* (AR). Già usata in ambiti molto specifici come nell'ambito militare e medicale o nella ricerca accademica, nel 2009 grazie al miglioramento della tecnologia, la realtà aumentata è arrivata al grande pubblico soprattutto attraverso un numero sempre crescente di applicazioni per qualsiasi tipologia di dispositivo. Tali applicazioni interagiscono attraverso un sistema di visualizzazione (webcam o fotocamera) con l'ambiente circostante.
Un esempio di applicazione di realtà aumentata disponibile gratuitamente per gli ormai sempre più diffusi *smartphone* è "*Layar*". *Layar* permette di puntare la fotocamera del proprio dispositivo portatile come se stessimo per scattare una foto e, ove disponibili, potrà visualizzare informazioni di ogni tipo sul "pezzo" di realtà che ci circonda, dalle fermate dei mezzi pubblici più vicine a prezzi e recensioni di ristoranti, film in sala e quant'altro.

Le applicazioni *Mixed Reality* (MR) fondono entrambi i concetti, creando ambienti all'interno dei quali oggetti reali ed oggetti virtuali coesistono ed interagiscono in tempo reale (figura 1.1).

Queste applicazioni ancorano i processi di renderizzazione secondo un sistema di riferimento generale, piuttosto che nel sistema di riferimento di un singolo ascoltatore, come nel caso della realtà virtuale. Il grado di immersione e la definizione di riferimento spaziale, sono caratteristiche connesse al concetto di virtuality continuum per display di visualizzazione introdotto da Milgram *et. al* [4]. Il *Continuum Virtuality* è una frase utilizzata per descrivere un concetto in cui esiste una scala continua compresa tra il completamente virtuale e la realtà.

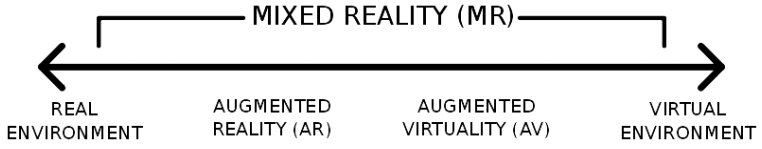


Figura 1.1: *Mixed Reality*

Queste nozioni possono essere adattate ai *Virtual Auditory Display* (VAD) e alla realtà audio aumentata (*Agumented Audio Reality*), includendo effetti sonori e sovrapponendo i suoni generati da computer in cima all'acquisizione dei segnali audio in *real-time*.

1.1 Sintesi dei gesti

La realtà virtuale si suddivide principalmente in due settori. Da un lato viene creato un ambiente virtuale nel quale inserire un avatar[1] utilizzato per replicare le movenze di un essere umano (o qualsiasi altro essere vivente) sintetizzandole in modelli moto-sensoriali.
Dall'altro lato si segue lo studio di movimenti e gesti, effettuato utilizzando una combinazione tra dispositivi elettronici di acquisizione video e varie metodologie di cattura, il riconoscimento di tali gesti e l'estrazione di caratteristiche e proprietà di quest'ultimi.
La realtà virtuale è uno spazio creato attraverso un computer dove l'essere umano può visualizzare, manipolare ed interagire con un insieme di dati estremamente complessi. Schematizzando, una realtà virtuale è caratterizzata da:

- la presenza di spazio simulato da un computer, uno spazio della memoria nel quale l'utente è proiettato con gradi e modalità di immersività differenti;
- l'interazione uomo-macchina;
- la mole dell'informazione gestita.

I modelli moto-sensoriali dei gesti (in particolare dei gesti *sound-producing*) vengono studiati da due diversi punti di vista: vale a dire le "teorie del controllo motorio" e la "sintesi computazionale di avatar che riproducono gesti umani". Le teorie del controllo motorio hanno lo scopo di capire il gesto in base alla sua biomeccanica, mentre la sintesi computazionale mira a comprendere l'intera riproduzione di un gesto riprodotto da un avatar in un ambiente virtuale.

L'interesse principale legato allo studio del gesto musicale, da un punto di vista biomeccanico e da un punto di vista dei modelli di controllo moto-sensoriale, è fornire una comprensione dinamica dei meccanismi responsabili di gesti *sound-producing*, i quali possono poi essere applicati in diversi contesti, come computer animation, videogiochi, sistemi interattivi con agenti virtuali incorporati o robot umanoidi.

[1] L'avatar è un'immagine scelta per rappresentare una un essere umano in un ambiente virtuale

Ci si concentra dunque sulla sintesi di gesti realistici degli avatar. La sintesi dei gesti umani reali è infatti una delle più grandi e complicate sfide della *computer graphics*, l'elevata sensibilità della percezione visiva umana riguardo alla postura naturale e al movimento biologico infatti è difficilmente replicabile nel mondo degli elaboratori. Il controllo del movimento degli avatar, tuttavia, rende necessario trattare con la complessità del corpo umano, il quale è composto da 206 ossa, oltre 200 giunti e da tessuti molli.

Per ottenere una sintesi realistica, è generalmente adottato un approccio stratificato alla rappresentazione di personaggi umani, nella quale gli scheletri possono supportare uno o più strati come ad esempio i muscoli, il tessuto grasso, la pelle, e gli strati di abbigliamento. In ciò che segue, distinguiamo tra la modellazione di avatar, inclusa l'anatomia, la biomeccanica e i modelli muscolo-schelettrici, e la modellazione del controllo motorio che ne permette il movimento.

Solitamente gli scheletri sono rappresentati da strutture gerarchiche (alberi o grafi) composte da una serie di segmenti rigidi collegati tra loro da giunzioni meccaniche. La posizione relativa e l'orientamento delle articolazioni determina la posizione dei diversi segmenti. All'interno di questa struttura ad albero, possono essere identificate delle catene articolate che caratterizzano gli arti superiori, le gambe, o la colonna vertebrale. Per ogni specifica catena articolata, è possibile calcolare la posizione del punto finale grazie ad un calcolo iterativo delle coordinate che passa da un giunto ad un altro.
Si potrà successivamente apprezzare l'esempio di tracciamento dello scheletro di una qualsiasi persona effettuato dall'applicazione "*Localization in Controlled Space*" (LCS) descritta nel capitolo 3, la quale, utilizzando il dispositivo Kinect come strumento di acquisizione di immagini e dati, traccia uno scheletro semplificato composto da 20 punti di giunzione, interconnessi tra di loro attraverso 19 segmenti.

Durante l'animazione degli avatar, è necessario progettare dei controllori specifici per le cinematiche scheletriche (tenendo conto delle traiettorie cinematiche) o per le dinamiche scheletriche (tenendo conto dello sforzo). Poichè questi sistemi scheletrici sono entrambi ridondanti, le trasformazioni e le dinamiche cinematiche possono essere rappresentate da particolari mappature (definite *many-to-one*). In altre parole, ingressi multipli sono in grado di produrre lo stessa uscita, pertanto, il controllo può essere basato sulla trasformazione inversa. Le trasformazioni inverse offrono una soluzione di un problema sotto un altro punto di vista. Partendo dal presupposto di conoscere il punto finale di una traiettoria oppure la struttura finale di un sistema meccanico, è possibile procedere con i calcoli degli stati del movimento a ritroso, fino alla posizione iniziale della traiettoria o dello stato del sistema meccanico

Generalmente sono considerati due problemi inversi chiamati: *inverse kinematics* (IK) e *inverse dynamics* (ID). Il problema della cinematica inversa consiste nel determinare lo stato degli angoli articolari, creati dagli snodi del braccio, data la posizione finale del braccio desiderato (informazioni fornite dal sensore). A causa della ridondanza delle possibili posizioni che braccio può assumere, anche quando è specificato l'andamento temporale della posizione della mano, gli angoli delle giunzioni non possono essere determinati in maniera univoca. Per esempio, un

braccio con sette gradi di libertà impegnati nel muovere un dito lungo un percorso desiderato all'interno di uno spazio tridimensionale, può realizzare questo spostamento con diverse sequenze della postura del braccio. Il problema delle dinamiche inverse può consistere nel determinare le tensioni dei muscoli agonisti e antagonisti, quando è noto lo stato del sistema (angoli articolari). Anche quando è specificato l'andamento temporale degli angoli delle giunzioni, ci sono davvero un numero infinito di forme d'onda di tensione dei muscoli che fanno muovere il braccio.

I modelli per la sintesi di gesti *sound-producing* sono un elemento cruciale nella progettazione degli avatar, essi infatti permettono di simulare movimenti che rispecchiano quelli della vita reale. Inoltre, prendendo in considerazione tutti i dati disponibili che accompagnano il movimento di un avatar, viene senza dubbio agevolato lo studio rivolto al miglioramento delle prestazioni reali di movimento.

Se ad esempio viene considerata un'esecuzione musicale, la modellazione moto-sensoriale fornisce una miglior comprensione di come la fatica possa essere ridotta e le lesioni causate dagli sforzi neuromuscolari possano essere evitate. Una comprensione più profonda del controllo moto-sensoriale può anche agevolare la formazione delle prestazioni per i musicisti, permettendo loro di migliorare la tecnica di esecuzione in base alle loro specifiche capacità biomeccaniche. I due principali approcci per la sintesi dei gesti sound-producing e dei gesti realistici in generale, sono "*data-driven animation*" e "*model-driven animation*".

Data-driven synthesis

I sistemi di motion capture, approfonditi nella sezione 1.2, forniscono una registrazione dei gesti di artisti reali e consentono uno sviluppo della sintesi basata sulla raccolta dei dati. I gesti per la maggior parte del tempo sono rappresentati dalla posizione e dall'orientamento dei marker situati nel corpo dell'artista. Sulla base di tali informazioni, lo scheletro può essere ricostruito in ogni periodo di campionamento.
La sintesi basata sulla raccolta dei dati sostanzialmente combina i valori forniti dalla motion capture con un database contenente una serie di posture e movimenti predefiniti. Grazie ad una interpolazione statistica rende possibile la riproduzione di gesti altamente realistici, con costi computazionali relativamente bassi.

Tuttavia, lo svantaggio principale è la mancanza di flessibilità. Infatti, rimane difficile modificare i movimenti registrati mantenendo il realismo del gesto. Tenendo conto della grande varietà dei gesti umani, è assurdo pensare di generare numerose tipologie di movimento dal momento che esiste una vastità di gesti generati in contesti differenti e con vari stili espressivi. Un altro limite tutt'altro che banale è l'adattamento dei movimenti esistenti con avatar di diverse morfologie. Due caratteristiche della sintesi di un gesto umano sono comunque molto importanti affinchè esso possa essere definito attendibile: l'occupazione dello spazio circostante, e la fluidità di movimento. Esistono pertanto studi volti a migliorare la naturalezza del movimento ricostruito e a confrontare le diverse tecniche di rendering, altri studi invece mirano a ridurre il numero di sensori e di marker durante la cattura dei movimenti di un utente, applicando vincoli pertinenti

alla postura dell'avatar.
Gli studi dei gesti associati alla musica sono per la maggior parte concentrati sull'analisi piuttosto che sulla sintesi.

Model-driven synthesis

L'approccio moto-sensoriale presuppone che esista una relazione tra le informazioni sensoriali e il comando del motore, tenendo conto della funzione da compiere (ad esempio il lancio di una pallina). Le informazioni sensoriali, osservate da specifici sensori, includono informazioni visive (la percezione visiva del gesto e l'ambiente), informazioni propriocettive (stato del sistema biomeccanico), e informazioni uditive. Il controllo motorio determina lo stato del sistema in ogni momento, ed è calcolato attraverso cinematiche inverse o attraverso processi dinamici inversi, o entrambi, a seconda della natura del modello di avatar (cinematica e dinamica). In questo modello, il compito è rappresentato in uno spazio omogeneo allo spazio di osservazione e che include informazioni sensoriali (ad esempio un obiettivo o una sequenza di obiettivi nello spazio visivo).

Per i programmatori che lavorano nel campo dell'animazione 3D, i problemi inversi sono di grande importanza. È molto più semplice infatti esprimere un gesto in termini di traiettorie spaziali con un punto finale piuttosto che esprimerlo in termini di angoli articolari o coppie. Tuttavia, l'avatar è composto da molte componenti articolate, quindi esistono un numero infinito di soluzioni per i problemi inversi. Inoltre, questi avatar contengono sia elementi passivi che attivi: gli elementi passivi possono essere caratterizzati da parametri meccanici come l'inerzia, la rigidità e la viscosità, mentre gli elementi attivi sono responsabili per l'attivazione dei muscoli. Per tali sistemi, non esiste un metodo generale per la progettazione di un controller in grado di gestire i vari parametri del sistema.
Un possibile approccio si basa quindi sulla definizione dei processi inversi con vincoli specifici, che identificano le migliori tra la vasta gamma di possibili soluzioni.

Tra tutti gli approcci proposti dagli studiosi in materia, l'architettura per la sintesi dei gesti *sound-producing* più interessante è composta sia dall'animazione di un avatar e sia dalla simulazione di uno strumento fisico adeguatamente modellato. Da un punto di vista funzionale, l'architettura è suddivisa in tre parti principali:

I. La prima parte contiene una banca dati di motion capture predefinite. Alcune serie temporali estratte dalle motion capture vengono anzitutto analizzate e poi utilizzate per guidare il controller moto-sensoriale.

II. La seconda parte è il controller moto-sensoriale stesso, il quale è composto da un modulo predisposto per cinematica inversa e una dinamica inversa basata su attuatori classici. L'intera unità di controllo moto-sensoriale aziona le dinamiche del modello umano attraverso la descrizione di forze e dei momenti applicati sulle specifiche articolazioni.

III. La terza parte è lo strumento di simulazione, il quale è dedicato alla simulazione di oggetti vibranti (ad esempio un tamburo). Sia l'avatar che lo strumento sono visualizzati in tempo reale, grazie ad un motore di rendering 3D.

Alla luce di tutto ciò, si può affermare che la realtà virtuale, ed in particolare la sintesi di un gesto di un avatar in una scena 3D, oltre ad essere uno strumento in continua evoluzione molto potente per lo sviluppo un'ampia gamma di attività, offre anche un ambiente potenzialmente ricco per lo studio e l'analisi della sintesi di un movimento.

1.2 Analisi del gesto

Gli esseri umani sembrano avere pochi problemi nel percepire e comprendere l'espressione di gesti di artisti musicali e dei ballerini su una scena. Anche quando non siamo in grado di vedere tutti i dettagli dei movimenti degli artisti, per esempio a causa della scarsa illuminazione, occlusione, o distanza, di solito percepiamo velocemente il carattere espressivo di tali movimenti, cioè se sono lenti, veloci, calmi, agitati.
Se per esempio alcune parti del corpo di un artista o ballerino in movimento sono momentaneamente occluse a causa di un angolo di visione sfavorevole, percepiamo che molto probabilmente esso continua a muovere tutto il corpo anche se non abbiamo effettivamente la possibilità di vederlo per intero. La nostre abilità di percepire molto accuratamente sia i movimenti reali che le loro caratteristiche espressive ed emotive, diventano ancora più notevoli se si cerca di replicare queste abilità con le macchine.

Quel che è facile per noi dunque può essere molto complicato o addirittura impossibile per strumenti basati su sistemi visivi, ma lo sviluppo di tecnologie per questa tipologia di macchine e il riconoscimento dei gesti ha attirato un notevole sforzo, in quanto tali sistemi artificiali di visione e di riconoscimento possono avere molte applicazioni nel campo della *Human-Machine Interaction* (HMI) utili per studiare una semplificazione, un'ottimizzazione o una alternativa ai vari compiti quotidiani e/o nel settore multimediale. Questa tecnologia infatti può dirci di più riguardo alle nostre capacità umane di visione e riconoscimento dei gesti.

1.2.1 Tecniche di motion capture

Dispositivi come semplici telecamere possono fungere da sensori per catturare i gesti umani.
Il processo di estrazione dei gesti utilizzando le immagini catturate dalla telecamera è simile a quello utilizzato da sensori tecnologicamente differenti. In primo luogo un sensore, ad esempio una macchina fotografica, misura un segnale, dopodiché il segnale viene elaborato e ne vengono estratte una serie di caratteristiche. Queste caratteristiche possono essere viste come una rappresentazione compatta del segnale di ingresso. In base al segnale d'ingresso, se ce ne fossero, vengono riconosciute le caratteristiche presenti nel gesto registrato e successivamente trasformate in un simbolo.

Quando un sistema è in esecuzione si genera una stringa di simboli che possono poi essere utilizzati per controllare la produzione musicale o anche qualcosa di completamente diverso. Quando il sensore è una fotocamera, il processo di estrazione delle caratteristiche si chiama *computer vision* e il processo di riconoscimento dei gesti è chiamato *pattern recognition*.
L'idea è di catturare il soggetto da più telecamere contemporaneamente. Una delle tecniche più utilizzate in passato consiste nel piazzare dei marker nelle posizioni del corpo più significative di un utente, come ad esempio le giunture (gomiti, ginocchia). I cosiddetti marker sono oggetti composti da un materiale che riflette la luce, in modo da creare un forte contrasto luminoso durante la cattura dell'immagine e facilitare così la cattura e il calcolo della posizione dei punti nei quali sono posizionati. I marker possono essere identificati in tutte le immagini e sfruttando una tecnica chiamata triangolazione (figura 1.2), si ottengono dei dati tridimensionali pronti per essere elaborati.
Se conosciamo la posizione relativa tra le due telecamere poi abbiamo due segmenti tridimensionali nello spazio e la loro intersezione ci permette di calcolare la posizione del marker rispetto alle telecamere.

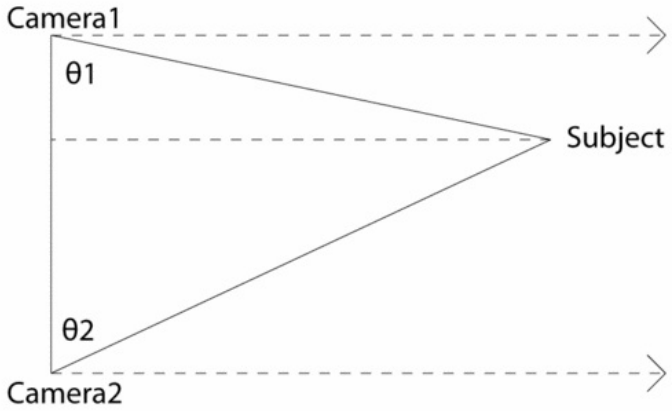


Figura 1.2: *Schematizzazione della tecnica di Triangolazione*

I marker possono assumere molte forme e dimensioni diverse e possono essere costruiti con materiali diversi. Esistono marker molto piccoli di forma sferica da applicare al volto umano utilizzati per il riconoscimento delle espressioni, marker passivi e attivi, quest'ultimi costituiti da un LED luminoso. L'approccio più comune è quello di utilizzare oggetti di forma sferica (per evitare dipendenze direzionali) verniciati con della vernice altamente riflettente simili a quelli visualizzati in figura 1.3.
I sistemi in cui vengono utilizzati questi marker sono spesso costruiti in modo tale che l'immagine sia molto scura ed in contrasto con i marker, i quali sono molto luminosi. Tali immagini sono quindi spesso riferite ai *moving light display* (MLD) o ai *point light display* (PLD).

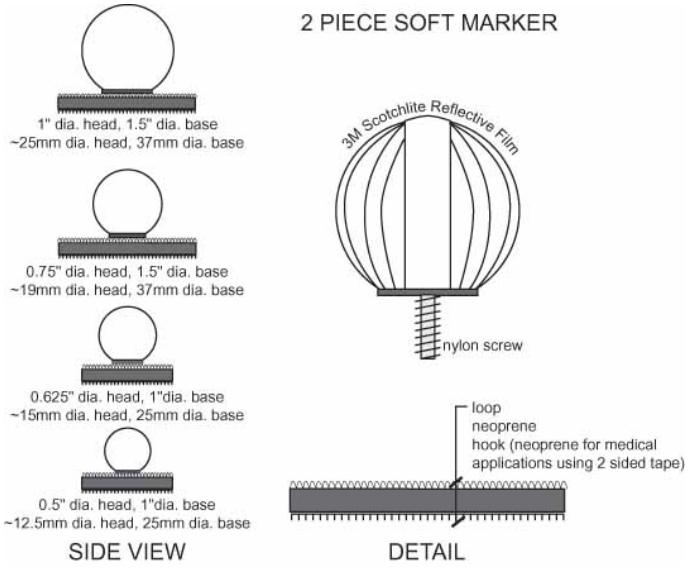


Figura 1.3: *Marker di varie dimensioni utilizzati per la motion capture*

L'utilizzo dei marker risale a più di 100 anni fa e, insieme alla tecnica della triangolazione, è tuttora in uso nella tecnologia motion capture (MoCap). Nel corso degli ultimi 10-20 anni sono nati diversi studi inerenti alla MoCap, questi studi hanno contribuito in maniera notevole alla ricerca in materia di scienza della salute, sport e divertimento. Per esempio nello sport, la MoCap è stata utilizzata per catturare il movimento di ogni arto di un atleta durante un salto in alto. Successivamente i dati raccolti sono stati associati ad un modello tridimensionale, il che significa che il salto può essere tranquillamente osservato da ogni possibile angolazione. Tutto ciò rappresenta un enorme passo avanti rispetto alla semplice registrazione video del salto.

La MoCap può essere utilizzata anche per analizzare la cinematica e la cinetica del movimento. In molti dei grandi film di Hollywood, i dati raccolti vengono utilizzati in corrispondenza alla computer grafica, per generare attori virtuali e molti altri effetti speciali. Per esempio, nel ben noto film "Jurassic Park", molti dei movimenti dei dinosauri si basavano sui dati provenienti dalla motion capture applicata agli elefanti. Analogamente nel film "Il Signore degli Anelli", i movimenti di un attore (Andy Serkis) sono stati catturati e utizzati per controllare la creatura Gollum (figura 1.4).
Tuttavia, in alcuni ambienti non è auspicabile (o possibile) l'utilizzo dei marker, come ad esempio nella videosorveglianza e nell'interazione uomo-computer. Nella videosorveglianza per esempio, saranno ricercate per lo più azioni anomale e/o ostili, mentre nelle applicazioni HMI, il sistema può cercare gesti particolari che sono definiti in anticipo e che sono operativi in un ambiente controllato. Un ambiente controllato è un ambiente virtuale nel quale ogni azione o avvenimento è monitorata, controllata e gestita dall'applicazione. La produzione musicale basata sul riconoscimento visivo dei movimenti può essere considerata come un caso particolare di interazione HMI. Ad esempio si pensi al riconoscimento e allo studio dei gesti effettuati da un

direttore d'orchestra, in base ai quali corrispondo delle variazioni musicali.

Figura 1.4: *Cattura dei movimenti dell'attore Andy Serkis*

1.2.2 Estrazione delle caratteristiche

L'estrazione delle caratteristiche ha lo scopo di convertire l'immagine di un gesto umano in una rappresentazione simbolica. I simboli sono derivati dalle caratteristiche, dove ogni caratteristica caratterizza alcuni aspetti del gesto, come il movimento o la sua forma. L'estrazione delle caratteristiche è un requisito generale di qualsiasi sistema basato su *computer vision*. Molti dei metodi di riconoscimento del modello che vengono applicati al riconoscimento visivo del gesto sono di carattere generale, dunque possono essere applicati al riconoscimento dei gesti utilizzando una qualsiasi tecnologia dei sensori (l'attenzione sarà focalizzata su metodi di estrazione non invasivi che non richiedono l'utilizzo di alcun tipo di marker).

Ci si concentrerà solo su caratteristiche relativamente semplici (o di basso livello) che possono essere estratte direttamente dall'immagine. Per esempio, il movimento generale di un corpo umano può essere rappresentato come il movimento del centro di massa del corpo. Il movimento di tutto il corpo è quindi ridotto al movimento di un singolo punto. Al contrario, per caratteristiche di alto livello, come l'estrazione di angoli articolari dai movimenti del corpo, richiede un'elaborazione più complicata. L'attenzione è rivolta ad un livello più basso di estrazione perché le funzioni sono relativamente veloci e facili da calcolare e, allo stesso tempo, sufficienti per la definizione di una ricca serie di gesti utili per il controllo. I metodi di estrazione delle caratteristiche saranno dunque raggruppati in base al tipo di informazioni su cui l'immagine si basa. Questi gruppi sono: *Silhouette-Based features*, *Apparence-Based features*, e *Motion-Based features*.

Silhouette-Based Features

Il primo passo da effettuare per applicare il metodo di estrazione *Silhouette-Based features* è estrarre la silhouette della persona umana dall'immagine in input e il secondo passo è quello di calcolare alcune caratteristiche dalla silhouette, come ad esempio la delle mani, del centro di massa del corpo, dei piedi e della testa (la posizione della testa è la caratteristica che verrà suc-

cessivamente trattata con maggior attenzione). La tecnica più utilizzata si basa sul presupposto che lo sfondo è statico. Questa ipotesi è realistica per ambienti interni illuminati da una piccola quantità di luce, ma per scene all'aperto o scene di interni più complessi vengono applicano metodi diversi. Dato che lo sfondo è statico, si può semplicemente sottrarre l'immagine di una persona in arrivo dall'immagine di sfondo e ottenere il primo piano della silhouette.

Purtroppo l'estrazione delle caratteristiche non sempre è esente da errori. Il sensore della macchina fotografica impreciso, piccole fluttuazioni della luce, l'abbigliamento indossato uomo molto simile allo sfondo sono motivi di errore. Questi errori si chiamano rumore.
Il rumore avrà un'influenza negativa sulla qualità delle caratteristiche che stanno per essere estratte, pertanto, è necessario rimuovere il rumore (se possibile), usando determinate tecniche di filtraggio.

Dopo aver estratto il profilo e rimosso gli eventuali errori causati dal rumore, si procede con il calcolo delle caratteristiche. La caratteristica più semplice si chiama centro di massa ed è rappresentata da due numeri. Il primo numero indica la posizione media orizzontale nell'immagine della sagoma umana e si calcola semplicemente sommando tutte le coordinate orizzontali dei pixel della silhouette e dividendo per il numero totale sempre dei pixel della silhouette. Il secondo numero è calcolato allo stesso modo, utilizzando però le coordinate verticali.
Anche se la caratteristica del centro di massa è molto semplice da estrarre, essa può essere molto potente visto che questa funzione può essere calcolata in maniera piuttosto robusta e molto veloce.

Una versione leggermente più complicata è quella di suddividere la silhouette in un certo numero di regioni e quindi calcolare il centro di massa di ogni regione. Questa estrazione più complessa può effettivamente consentire una approssimazione grossolana riguardo alla posizione degli arti dell'utente.

Un'altra semplice caratteristica è il rettangolo di selezione della silhouette. Un riquadro è il rettangolo più piccolo che racchiude la silhouette. Osservando i mutamenti della sua larghezza e altezza nel tempo, possono essere riconosciute alcune azioni. La piattaforma free software "EyesWeb" (`www.eyesweb.org`) contiene un modulo che delimita i segmenti del riquadro di in piccoli sotto riquadri, i quali corrispondono grosso modo, in alcune posizioni, alle varie parti del corpo.
Esiste un altro tipo di caratteristica analoga al rettangolo di selezione, solo che a differenza del rettangolo utilizza il più piccolo cerchio o ellisse che contiene la silhouette e poi utilizza i parametri di tale cerchio o ellisse come caratteristiche per ulteriori elaborazioni.

Apparence-Based Features

Se un sistema è in grado di estrarre la posizione della testa e delle mani da una immagine di una silhouette, allora può essere sufficiente per il riconoscimento di una serie di gesti. Tuttavia, in alcune situazioni non è possibile estrarre la posizione di queste caratteristiche a causa della

configurazione (ad esempio uno sfondo non statico), di vestiti particolarmente ingombranti, o a causa di una risposta in tempo reale. Pertanto, in alternativa, possono essere applicati dei metodi chiamati *Apparence-Based*.
Essi si basano sull'idea che l'utente ha un colore differente rispetto allo sfondo. Da un lato si assume uno sfondo monocromatico, ad esempio verde, il colore verde sarà un colore che l'utente non indossa. Effettuando dunque la rilevazione di tutti i pixel non verdi può essere trovata rapidamente la silhouette degli utenti.

Questo principio si chiama *Chroma-Key* (figura 1.5) e viene utilizzato molto spesso in ambiti differenti, per esempio nelle previsioni meteorologiche televisive. Dall'altro lato, il sistema trova mani e viso dell'utente effettuando una ricerca all'interno dell'immagine dei pixel con lo stesso colore della pelle, per poi raggrupparli.

Figura 1.5: *Tipica applicazione del processo Chroma-Key con sfondo verde*

Un pixel colorato in una fotocamere è rappresentato da una quantità di colore rosso (R), una quantità di colore verde (G), e una quantità di colore blu (B). Questa rappresentazione è chiamata RGB. Supponiamo di avere otto bit per rappresentare ogni colore, allora si possono calcolare 256 diverse tonalità per ogni colore.
Combinando i tre colori si possono ottenere 16.777.216 colori diversi. Il quantitativo di colore di un pixel è solitamente scritto come RGB [123,45,76] (in questa terminologia un pixel rosso massimo è RGB [255,0,0], uno completamente bianco è RGB [255,255,255], e un pixel completamente nero è RGB [O, O, O]). A questa rappresentazione si aggiunge una rappresentazione chiamata *Hue-Saturation-Intensity* (HSI), la quale descrive i valori di intensità, tonalità e saturazione del colore.

Il rilevamento dei pixel dell'oggetto viene effettuato confrontando i valori dei pixel in arrivo con i valori HSI desiderati. Per esempio viene impostato un oggetto con valori di tonalità compresi nell'intervallo [50,80], allora tutti i pixel in arrivo dall'immagine con una tonalità all'interno di questo intervallo saranno classificati come appartenenti all'oggetto e quelli esterni apparterranno allo sfondo.
Dopo aver trattato il rumore nell'immagine di uscita (usando gli stessi metodi descritti precedentemente) possono essere estratte le caratteristiche. L'estrazione può essere assai complicata come è avvenuto nella precedente descrizione del metodo *Silhouette-Based*, ma di solito viene estratto solo il centro di massa.
L'idea del rilevamento della pelle è affascinante, ma può diventare una operazione molto complessa nei sistemi reali, pertanto al metodo *Apparence-Based* per il riconoscimento dei gesti viene spesso associato l'uso dei marcatori.

Motion-Based Features

Il metodo di estrazione delle caratteristiche chiamato *Motion-Based* si basa sullo stesso principio del metodo *Silhouette-Based*, e cioè che lo sfondo sia statico, utilizza infatti una sottrazione dello sfondo dalle immagini molto simile, con l'unica differenza che lo sfondo non viene acquisito dall'immagine in arrivo, ma l'immagine in ingresso viene sottratta dall'immagine precedente, in modo da ottenere due silhouette differenti, la sottrazione delle due fornisce dei dati riguardo al movimento dell'utente. Le caratteristiche vengono quindi estratte esclusivamente quando l'utente è in movimento.

1.2.3 Mappatura delle caratteristiche

Le caratteristiche che vengono estratte da un'immagine o da un flusso di immagini possono essere ulteriormente trasformate in simboli o parametri che fungono da segnali di controllo per la musica interattiva e sistemi multimediali.
Il processo generale di mappatura delle caratteristiche in simboli o parametri si chiama *pattern recognition*, ma quando si concentra sul movimento umano, il modello diventa un gesto e l'approccio è chiamato *gesture recognition*. Nei gesti utilizzati per il controllo dei suoni, sono spesso applicate funzionalità e metodi di riconoscimento relativamente semplici. Caratteristiche e metodi più complicati infatti, possono implicare un calcolo troppo complesso per l'elaborazione dei dati in tempo reale. Tuttavia, caratteristiche e semplici metodi di riconoscimento spesso producono risultati poco (o per niente) accurati. Un'altra caratteristica dei metodi comunemente utilizzati per il riconoscimento dei gesti basati sul controllo sonoro è che tali metodi spesso operano su dati statici, cioè sul riconoscimento dei gesti in una singola immagine, esattamente il contrario di ciò che accade durante il riconoscimento dei gesti basato su una serie di immagini consecutive.
Indichiamo questi due tipi di riconoscimento come *static* e *dynamic gesture recognition*.

Static gestures recognition

Il centro di massa può essere utilizzato come parametro di controllo per il volume e l'altezza di un suono, e, in una versione più avanzata, questo può essere combinato con una serie di regioni nell'immagine in modo che le caratteristiche estratte da ogni controllo della singola regione corrispondano ad uno o più parametri del sistema audio.

In realtà, un vero e proprio sistema di riconoscimento dei gesti dovrebbe mappare le caratteristiche in uno o più simboli, come ad esempio acceso o spento. Per esempio, l'immagine può essere suddivisa in un certo numero di regioni e la quantità di movimento in ogni regione può essere calcolata e confrontata con un preciso valore di soglia. Se la quantità di movimento è al di sopra della soglia si può concludere che una certa attività è presente in questa regione e quindi può essere utilizzata per controllare un evento nel sistema audio.
Tali eventi possono essere ulteriormente parametrizzati utilizzando i valori reali di una o più caratteristiche.

L'idea sta nel creare un modello per ogni gesto, le caratteristiche dell'immagine in ingresso vengono poi confrontate con tutti i modelli, e il modello più simile a quello d'ingresso viene identificato come gesto riconosciuto.
Per esempio, se cinque caratteristiche rappresentano un gesto, questo gesto può essere rappresentato come un punto in cinque dimensioni dello spazio. Ogni immagine in arrivo è un nuovo punto in questo spazio a cinque dimensioni, il riconoscimento consiste nel calcolare la distanza tra il nuovo punto e i punti che rappresentano il possibile gesto.

Ovviamente, ci sarà sempre qualche diversità nel modo in cui i diversi utenti eseguono un gesto, inoltre, il rumore dell'immagine può anche causare una differenza tra le caratteristiche che rappresentano tale gesto. La conseguenza è che un modello non può effettivamente essere rappresentato semplicemente da un punto, ma deve essere rappresentato da un numero di punti nello spazio a cinque dimensioni.
Una serie di immagini contenenti un gesto possono essere analizzate al fine di catturare questa diversità.
Quando si esegue il riconoscimento dei gesti statici si devono sempre considerare due aspetti, cioè, il rumore e il problema chiamato in gergo *Midas Touch*.

Molti sistemi contengono un rumore, con la conseguente incertezza nell'estrazione delle caratteristiche. Un gesto statico quindi è spesso riconosciuto solo dopo essersi presentato per un certo numero di immagini consecutive o utilizzando un filtro nel flusso dei dati in uscita.

Un problema diverso con una soluzione relativa è il problema *Midas Touch*. L'espressione "*Midas Touch*" deriva da un antico mito greco, il quale narra di un re che trasforma in oro tutto ciò che tocca. Inizialmente è tutto meraviglioso, ma dopo poco tempo re Mida si accorse che detenere questo potere non è poi così bello, infatti non poteva mangiare visto che non riusciva a toccare cibo senza trasformarlo in oro.

Una situazione analoga è presente nel riconoscimento dei gesti. Per esempio, come si fa a distinguere se un utente sta semplicemente muovendo la mano o se sta facendo un gesto?
Normalmente un sistema di riconoscimento del movimento è impostato per classificare ogni immagine in ingresso, analizzando i possibili gesti che essa contiene. Tutto ciò può essere equivoco nel momento in cui l'utente compie involontariamente un movimento. La soluzione spesso consiste nell'includere un filtro temporale, anche se tale metodo rallenta il riconoscimento del gesto durante l'operazione di cattura e traduzione.

Dinamyc gestures recognition

Il riconoscimento dinamico dei gesti si basa su metodi che utilizzano alcuni filtri nella successione di immagini. Un gesto può quindi essere definito come una transizione da una configurazione ad un'altra.
Prendiamo come esempio un pollice di una mano. Se il pollice si trova ad essere vicino alla mano in un'immagine seguita da una seconda nella quale è lontano dalla mano, allora viene riconosciuto un gesto associabile ad un "click".
Per il riconoscimento di gesti dinamici è necessario che ci sia un punto di inizio e uno di stop, dal momento che l'estrazione produce una lunga serie di caratteristiche. Tutto ciò deve essere suddiviso in ulteriori sequenze, ognuna delle quali rappresenta un gesto individuale. Non esiste alcuna soluzione standard a questo problema e quindi molti sistemi utilizzano una soluzione ad hoc.

Un secondo problema riguarda il fatto che utenti differenti potrebbero eseguire un gesto più o meno rapidamente rispetto al modello che rappresenta un gesto. Ciò renderà la traiettoria più o meno lunga e di conseguenza più difficile il riconoscimento.
Un modo per ovviare a questo problema è quello di applicare un *Dynamic Time Warp*[2] che in qualche modo si occupa di questo problema.

Un altro approccio è quello di dividere la traiettoria in piccoli pezzi che possono essere riconosciuti in modo indipendente e uniti in un secondo momento in modo da riconoscere l'intero gesto. In alternativa, può essere utilizzato il modello *Hidden Markov* (HMM).
HMM è un approccio statistico nel quale ogni gesto è rappresentato da una sequenza degli stati legati da probabilità di transizione, cioè la probabilità di uno stato nell'immagine corrente dato un altro stato nell'immagine precedente. Il modello HMM prende una sequenza di input e calcola quale gesto è più probabile a quello riportato in ingresso.

1.3 Audio 3D

Dopo un'approfondita descrizione sulla sintesi e la cattura dei gesti, sull'estrazione delle caratteristiche e sul riconoscimento di un gesto, l'attenzione viene focalizzata sull'importanza del

[2] algoritmo per misurare la somiglianza tra due sequenze che possono variare nel tempo o nella velocità

tracciamento della testa, che rappresenta una importante fonte di informazioni per perseguire lo studio di applicazioni di riproduzione sonora binaurale e renderizzazione audio 3D.

Il problema da affrontare è creare un modello che può essere impiegato per coinvolgere la riproduzione sonora oltre a differenti gradi della virtualità. Ci si concentra su sistemi basati sulla riproduzione sonora in cuffia per un Binaural Rendering[3], tenendo conto che possibili svantaggi (come ad esempio l'invasività delle cuffie o una risposta in frequenza instabile) possono essere accompagnati da una serie di caratteristiche auspicabili. Questi sistemi infatti eliminano il riverbero e altri effetti acustici che derivano da un qualsiasi ambiente di ascolto, riducendo il rumore di fondo e fornendo un monitoraggio audio adattabile, i quali sono tutti aspetti rilevanti, specialmente in un contesto che possiede dei margini di miglioramento.

Con questo tipo di sistema ogni orecchio riceve segnali distinti, semplificando notevolmente la progettazione di tecniche di renderizzazione 3D del suono.
Al giorno d'oggi la maggior parte dei sistemi *mixed reality* (MR) sono in grado di controllare in maniera fluida due dimensioni dello spazio uditivo, cioè controllare delle sorgenti sonore posizionate sul piano orizzontale riferendosi ad un sistema di coordinate centrato sulla testa (*head-centric*). Le tecnologie di tracciamento della testa insieme alle risposte in frequenza *dummy-head*[4] o a modelli adattabili per la localizzazione orizzontale permettono una discriminazione accurata tra sorgenti sonore disposte intorno all'utente e nel sottospazio definito.
La terza dimensione, l'elevazione o dimensione di controllo verticale, richiede una specifica caratterizzazione utente in modo da simulare la percezione efficace sul piano verticale dovuto principalmente alla forma esterna dell'orecchio (il padiglione auricolare). I compiti fondamentali necessitano di trovare un modello adatto per descrivere il contributo padiglione auricolare con l'estrazione dei parametri relativi alle misure antropometriche.

1.3.1 Sistema di riferimento

Una scena audio 3D, creata da una riproduzione del suono binaurale, sarà stimata da ogni singolo segnale di sorgente sonora utilizzando i relativi metadati, dopodiché per produrre il segnale stereo inviato agli auricolari si sommano il segnale destro e il segnale sinistro. Questa architettura può anche consentire una variazione di scala efficace in funzione delle risorse computazionali disponibili o della larghezza di banda.
Alcuni criteri psicoacustici possono definire una priorità di renderizzazione delle sorgenti sonore e di alcune caratteristiche, come ad esempio l'udibilità della sorgente. In particolare, in relazione alla quantità di banda disponibile, fonti meno percettibili possono essere rimosse dalla scena e questa accurata degradazione della scena renderizzata si tradurrebbe in una esperienza soddisfacente anche nei casi di scarsa qualità del servizio.

Nelle tipiche applicazioni virtuali audio la testa dell'utente è il riferimento centrale per l'au-

[3] Ambiente sonoro adatto alla percezione binaurale (a due orecchi).
[4] Risposte in frequenza riferite ad una testa fittizia (la traduzione di *dummy* corrisponde a manichino).

dio il rendering degli oggetti. In linea di principio la posizione della testa dell'utente stabilisce un sistema virtuale di coordinate e costruisce una mappa della scena uditiva virtuale. Nel caso di un ambiente misto, gli oggetti sonori sono posti nello spazio fisico intorno all'utente e quindi, concettualmente, posizionati in coerenza con un sistema di coordinate fisiche. La localizzazione di oggetti audio virtuali in un ambiente misto richiede la sovrapposizione di un sistema di coordinate rispetto ad un altro.
A seconda della natura della specifica applicazione, le diverse impostazioni possono essere utilizzate per classificare il sistema di coordinate per gli oggetti audio virtuali e la posizione degli oggetti nell'ambiente. Una semplice distinzione è la scelta di riferirsi ad un sistema di posizionamento globale, o ad un sistema di coordinate locali.
Una classificazione ideale può contribuire alla definizione delle possibili applicazioni che utilizzano le tecnologie di spazializzazione audio. In alcuni casi è necessario far corrispondere i due sistemi di coordinate in un modo che le sorgenti sonore virtuali appaiano in luoghi specifici nell'ambiente fisico, mentre in altre applicazioni le sorgenti virtuali sono flottanti da qualche parte intorno all'utente perchè il target si trova ad un livello concettuale disgiunto nell'interazione con l'utente.

Al fine di aiutare la comprensione, uno schema visivo di due diverse applicazioni è mostrato in figura 1.6. La caratterizzazione si muove intorno ad un semplice spazio bidimensionale definito in termini di *degree of immersion* (DI) e la *coordinate system deviation* (CSD) [3]. Il nostro punto di vista è una semplificazione dello spazio tridimensionale proposto da Milgram, composta da *Extent of World Knowledge* (EWK), *Reproduction Fidelity* (RF) e la *Extent of Presence Metaphor (EPM)*. Tutte le sigle sono riferite all'articolo "*Augmented reality: A class of displays on the reality-virtuality continuum*" [4]. Le corrispondenze sono tracciate con particolare attenzione ai tre soggetti coinvolti: il mondo reale, il motore MR e l'ascoltatore.

Il motore MR è l'intermediario tra la realtà e la rappresentazione percepita da chi ascolta, in questo senso CSD corrisponde con EWK. Un basso CSD significa un EWK alto: il motore MR ha cognizione di tutti gli oggetti presenti nella scena reale e può rendere la scena acustica sintetica, come se l'ascoltatore percepisse un mondo coerente. D'altra parte la questione del realismo riguarda le tecnologie coinvolte nel motore MR, e la complessità della tassonomia di tale sistema aumenta in modo considerevole. EPM e RF non sono del tutto ortogonali e la scelta presa è quella di definire DI secondo la seguente idea: quando un ascoltatore è circondato da un suono reale, tutto il suo corpo (di lui/lei) interagisce con le onde acustiche, ovvero una tecnologia ad alto tasso di realismo è in grado di monitorare per intero posizione ed orientamento del corpo dell'ascoltatore (alta DI). Tornando alla figura 1.6, il soggetto a sinistra fornisce un esempio di alta DI corrispondente ad una percentuale elevata del corpo, tracciata nel sistema virtuale di coordinate (un corpo quasi completamente colorato di grigio), e presenta una bassa deviazione delle coordinate virtuali dal sistema di coordinate fisiche, causata da una semplice traduzione. Al contrario il soggetto sulla destra presenta un basso DI e un elevato CSD, rappresentato da una testa grigia e da un ascoltatore centrato nello spazio virtuale 2D.

Esempi concreti dei due casi sono i seguenti. L'utente femminile indossa un dispositivo mo-

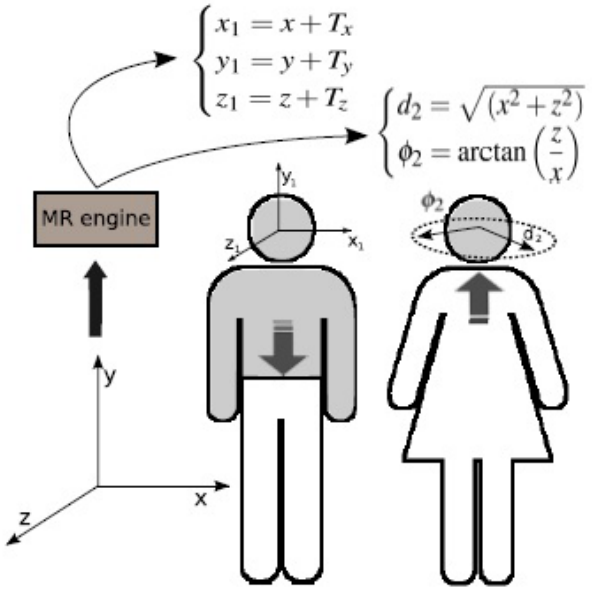


Figura 1.6: *Distinzione delle coordinate in un ambiente Mixed Reality*

bile e, grazie a quest'ultimo, è immersa in una moltitudine di messaggi e comandi. L'utente maschio è in un'operazione di telepresenza pericolosa, la sua testa e il suo tronco sono monitorati per immergere il suo corpo in un luogo reale lontano. Il primo scenario rappresenta un mondo totalmente virtuale e in molti casi non completamente tridimensionale, il secondo, invece, rappresenta la sovrapposizione esatta del mondo virtuale con quello reale.
Il caso più comune di un sistema di coordinate virtuale flottante è quello in cui l'unico punto di ancoraggio relativo all'evento localizzato è la testa dell'utente. Di solito, sorgenti sonore virtuali sono trasmesse da diverse direzioni e sono puramente virtuali (minimo DI e massimo CSD).

1.3.2 Riproduzione binaurale e head-realted transfer function

Le tecniche per l'individuazione di una sorgente sonora nello spazio, seguono diversi approcci. Una prima distinzione riguarda il metodo di riproduzione del suono, ovvero l'utilizzo di altoparlanti a differenza dei sistemi basati sull'utilizzo di cuffie (*headphone-based systems*).
L'applicazione LCS descritta nel capitolo 3 oltre a tracciare la posizione della testa di un utente, prevede la possibilità di spostare un punto di sorgente sonora nello spazio intorno alla persona.

Le tecniche binaurali si trovano all'interno dei due gruppi (la riproduzione binaurale può essere ottenuta o con gli altoparlanti o le cuffie) e consente esperienze uditive autentiche se i timpani sono stimolati da segnali sonori che mantengono all'incirca la stessa pressione che si riscontra in reali condizioni.

Due approcci alternativi riferiti solamente alla categoria di riproduzione degli altoparlanti sono: il tentativo di ricreare l'intero campo del suono su un'area di ascolto più ampia (ad esempio la tecnica di *Wave field synthesis*[5]) e l'intento di introdurre solo gli elementi di cui il sistema uditivo ha bisogno per stabilire tramite la percezione la posizione del suono (ad esempio *Ambisonics*[6]).

Tuttavia, l'uso basato sulla riproduzione in cuffia, in combinazione con i dispositivi di tracciamento della testa, garantisce un grado di interattività, realismo e immersione che non è facilmente realizzabile con sistemi multicanali o *Wave field synthesis*, a causa delle limitazioni nello spazio di lavoro dell'utente e di effetti acustici dello spazio reale di ascolto.

Le *head-realted transfer function* (HRTFs) catturano le trasformazioni subite da un onda sonora nel suo percorso dalla sorgente al timpano. Le trasformazioni accadono a causa della diffrazione e dalle riflessioni causate dalla testa, dai padiglioni auricolari, dal busto e dalle spalle dell'ascoltatore.
Tale caratterizzazione permette il posizionamento virtuale delle sorgenti sonore nello spazio circostante: in coerenza con la sua posizione rispetto alla testa di chi ascolta, il segnale viene filtrato attraverso la coppia corrispondente di HRTFs creando un segnale per l'orecchio destro e un segnale per l'orecchio sinistro per essere riprodotti dalle cuffie.
In questo modo possono essere simulati e integrati campi sonori a tre dimensioni con un elevato senso di immersione in contesti *mixed reality*. Tuttavia, la registrazione di HRTFs personalizzate per un singolo ascoltatore, richiede attrezzature specifiche, costose e delicati processi di trattamento audio. Questi elementi rendono difficile l'utilizzo delle HRTFs personalizzate in ambienti virtuali, considerando il costo elevato delle altre componenti di coinvolgimento (*motiontracker* e dispositivi tattili).

Per queste ragioni le funzioni di trasferimento generalizzate (cioè HRTFs per una *dummy-head*) sono utilizzate in alcune applicazioni, da cui ne conseguono evidenti errori di localizzazione del suono quali la percezione errata del livello di elevazione della sorgente sonora o inversioni fronte retro e la mancanza di esternalizzazione.

Una serie di esperimenti sono stati condotti da Wenzel *et. al* [2] al fine di valutare l'efficacia di funzioni di trasferimento generalizzate per la visualizzazione virtuale dell'acustica. Si ottiene una percezione della precisione angolare orizzontale sia in condizioni reali, sia con il rendering audio 3D, molto simile impiegando dell HRTFs generalizzate, anche se alcuni esperimenti dimostrano che l'uso di funzioni generalizzate aumenta il tasso di errori di inversione.

Anche Begault *et. al* [1], seguendo questa linea, ha confrontato l'effetto di una HRTF individuale e di una generalizzata, con *head-tracking* e riverbero applicato ad un suono vocale.

[5]textitWave field synthesis (WFS) è una tecnica di rendering audio spaziale, caratterizzata dalla creazione di ambienti acustici virtuali.

[6]Serie di tecniche di registrazione e riproduzione multicanale che utilizza la tecnologia di miscelazione, può essere utilizzata sia dal vivo sia in studio.

I loro risultati hanno mostrato che il tracciamento testa è fondamentale per ridurre gli errori dovuti dal'angolo e in particolare per evitare inversioni, mentre la percezione dell'azimut in una generica HRTF di ascolto è leggermente peggiorata se confrontata con funzioni di trasferimento individuali ed è equilibrata con l'introduzione di un riverbero artificiale.

L'*head-tracking* utilizzato in determinate applicazioni, rende superfluo l'utilizzo delle mani da parte dell'utente. Una tipica operazione è quella dello scrolling di una finestra generica di testo. Con una tecnica del tutto simile si può, con lo spostamento dello sguardo, muoversi attraverso più finestre a video, muovere le finestre stesse da una posizione ad unaltra o spostarsi all'interno di menu a tendina.
Il riconoscimento della posizione della testa (ed in particolare il riconoscimento degli occhi) è stato utilizzato per sviluppare dei sistemi di allarme integrati in prototipi di automobili. Tali sistemi individuano movimenti bruschi della testa oppure misurano la frequenza con cui il guidatore batte le palpebre, ed emettono particolari segnali acustici per evitare colpi di sonno al guidatore.

In sintesi, mentre le generiche HRTFs rappresentano un mezzo economico e semplice di fornire una percezione 3D nella riproduzione in cuffia, l'ascolto di generici suoni spazializzati può provocare errori di localizzazione del suono, il quale non può essere completamente controbilanciato da ulteriori *spectral cues*, soprattutto in condizioni statiche. In particolare, gli *elevation cues* non possono essere caratterizzati attraverso la generalizzazione delle caratteristiche spettrali.

In conclusione, oltre alla dipendenza critica sulla posizione assunta dall'ascoltatore in relazione alla sorgente sonora, le caratteristiche antropometriche del corpo umano hanno un ruolo fondamentale nella caratterizzazione delle funzioni di trasferimento *Head-Related*.

1.3.3 Applicazioni di realtà audio aumentata

Esiste una vasta gamma di applicazioni audio 3D adattabili alle tecniche di riproduzione binaurale.
A titolo di esempio, servizi di informazione come news, e-mail, eventi del calendario o altri tipi di messaggi, possono essere posizionati nello spazio virtuale acustico intorno alla testa dell'utente. Eventi di calendario, sotto forma di messaggi vocali, sono trasmessi da differenti direzioni a seconda degli impegni presenti nell'agenda dell'utente, e tarati in modo che mezzogiorno sia posizionato esattamente nella parte frontale dell'ascoltatore.
Nel caso di una presentazione della struttura gerarchica del menu, come si usa comunemente nei dispositivi mobili, può essere adottato il design dell'interfaccia spaziale dell'utente come quello presentato durante il "*4th International Symposium on Mobile Human-Computer Interaction*" tenutosi a Londra nel 2002.

Applicazioni di realtà virtuale coinvolgenti utilizzano anche specifici sistemi di coordinate virtuali, di solito collegate alla geometria di una scena grafica a realtà virtuale. Nelle applicazioni di gioco per computer che utilizzano tecniche di spazializzazione audio, il sistema di coordinate

virtuale viene definito in base alla scena di gioco e, a volte, combinate con le informazioni sulla posizione fisica di un utente (come ad esempio il tracciamento della testa via webcam).

La telepresenza è un altra tipologia di sistema flottante di coordinate virtuali ed è simile ai sistemi virtuali di monitoraggio uditivo. Un ulteriore esempio di realtà mista è la *bidirectional augmented telepresence application*, dove un segnale di telepresenza binaurale è collegato ad un ambiente pseudo-acustico.
Il motore MR fonde l'ambiente locale pseudo-acustico con un ambiente remoto dello stesso tipo. Presupponendo di porre due utenti in questi due ambienti, è possibile riprodurre a livello uditivo l'ambiente dell'altra persona. In questo caso il CSD legate all'ambiente remoto è molto basso.
In ambienti virtuali collaborativi, Benford ha dimostrato che spunti spaziali possono essere combinati a quelli audiovisivi in modo naturale per facilitare la comunicazione. Il ben noto "*effetto cocktail party*"[7] dimostra che le persone possono facilmente monitorare una serie di differenti flussi audio all'interno di uno spazio contemporaneamente, focalizzandosi sul suono ritenuto di maggior interesse.
Durante teleconferenze multicanali, il posizionamento di ciascun oratore può essere scelta liberamente all'interno di una sala riunioni virtuale.

Ricercatori come Walker e Brewster hanno esplorato l'utilizzo dell'audio spazializzato su dispositivi mobili, con lo scopo ad esempio, di risolvere i problemi di ingombro visivo in interfacce di dispositivi mobili. Questo potrebbe fornire un aiuto per eliminare l'ambiguità degli altoparlanti nel caso di convegni multicanali, e offrire un ulteriore approfondimento per movimenti basati su una spazializzazione audio aumentata durante scenari di teleconferenze multicanali nei dispositivi portatili.

Quando si posizionano audio-oggetti virtuali in luoghi stabiliti all'interno del mondo fisico intorno a un utente, il sistema di coordinate usato per il rendering dei suoni virtuali devono coincidere con una mappa dell'ambiente fisico. Sarebbe idealmente possibile inserire un audio-oggetto virtuale in prossimità di qualsiasi oggetto fisico del mondo reale.
Messaggi audio localizzati in prossimità di un opera d'arte esposta al museo ed introduzioni ad una mostra sono esempi di sistemi di audio guida: un Post-it acustico è legato ad un sistema di coordinate fisiche.
Un messaggio registrato viene ascoltato da un visitatore nel momento in cui egli è posto in una determinata posizione del museo. La posizione e l'orientamento degli utenti sono continuamente aggiornati e le caratteristiche acustiche dell'edificio sono sempre monitorate, ne risulta dunque un altissimo DI, così un paesaggio sonoro dinamico circostante e diversi messaggi vocali vengono riprodotti attraverso cuffie senza fili.
Le osservazioni di cui sopra sono particolarmente rilevanti per le applicazioni portatili e per un controllo mobile che non dipende dalla vista. Sistemi ausiliari alla navigazione per i non vedenti rappresentano uno spiccato utilizzo di queste tecnologie. In queste applicazioni la mappa dello

[7] L'effetto *cocktail party* avviene quando riusciamo a captare segnali acustici distinti che attirano la nostra attenzione, in presenza di una situazione caotica e rumorosa.

spazio fisico può essere globale o locale.

Due esempi conclusivi per i sistemi fisici di coordinate locali sono i display uditivo-virtuali per equipaggi aerei durante la simulazione delle missioni di volo e sistemi d'allarme di collisione per piloti di volo. In queste ultime applicazioni il sistema fisico di coordinate associato si muove con l'aereo e in entrambi i casi si ottiene un basso CSD, il che significa che la corrispondenza tra il sistema di coordinate fisico e il sistema di coordinate virtuale è il lavoro più critico.

Capitolo 2

Kinect: Controller e Motion Tracker

Microsoft Kinect (inizialmente conosciuto con il nome Project Natal), è un accessorio per Xbox 360 sensibile al movimento del corpo umano. Esso rende il giocatore stesso controller della console senza l'uso di alcuno strumento, a differenza dei concorrenti come Nintendo Wii o Sony Playstation.
Sebbene in origine il dispositivo Kinect fu pensato esclusivamente per Xbox 360, Microsoft prevede di rendere nel prossimo futuro disponibile l'uso della periferica ai PC dotati del nuovo sistema operativo Windows 8. Microsoft ha però dichiarato di rilasciare gratuitamente, durante l'estate 2011, i driver ufficiali per poter utilizzare Kinect nel proprio Personal Computer, dimostrando così di voler portare quanto prima la tecnologia di Kinect anche sui sistemi operativi Windows attualmente disponibili, favorendo lo sviluppo di varie applicazioni tra il mondo degli sviluppatori di software.

KINECT
for XBOX 360

Figura 2.1: *Microsoft Kinect.*

2.1 Storia

Kinect è stato annunciato al pubblico il 1 giugno 2009 durante la conferenza stampa della Microsoft all'E3 2009 (*Electronic Entertainment Expo*) con il nome Project Natal, poi rinominato Kinect alla presentazione ufficiale all'E3 2010. Il 13 giugno 2010 Microsoft ha rivelato per la prima volta il vero nome del dispositivo, ovvero Kinect. Quest'ultimo è in vendita dal 4 novembre 2010 in America e dal 10 novembre in Europa, ed è possibile usarlo su un qualsiasi modello di XBOX 360.

L'hardware di Kinect si basa su tecnologie della 3DV, una compagnia israeliana specializzata in tecnologie di riconoscimento dei movimenti tramite videocamere digitali che Microsoft ha prima finanziato e poi acquisito nel 2009, e sul lavoro della israeliana PrimeSense, che ha poi dato in licenza la tecnologia a Microsoft. Il software di Kinect è stato, invece, sviluppato internamente ai Microsoft Game Studios e, più precisamente, dai programmatori della Rare, la quale ha dovuto cancellare altri progetti migliori per dedicarsi interamente alla periferica.
L'uscita di Kinect ha provocato un grande sommovimento nella comunità di sviluppo libero di software per PC e Mac. Una moltitudine di programmatori è al lavoro sul *reverse engineering* sulla periferica, allo scopo di trovare nuove modalità di utilizzo di un dispositivo che si configura come il primo di una serie di sistemi che potrebbe davvero portarci ad un futuro alla *Minority Report*.

Tra le tante applicazioni che dall'uscita di Kinect hanno visto l'utilizzo non ludico della periferica Microsoft citiamo qui quella sviluppata da due ricercatori dell'universit di Berna (Svizzera). Questi hanno dimostrato un sistema di controllo per la visualizzazione e l'analisi di radiografie e tomografie che svincola i medici dall'uso delle mani e rende, grazie anche all'uso delle capacità di riconoscimento vocale di Kinect, davvero immediata la consultazione degli esami medici, anche durante un'operazione in corso.
Sono in sviluppo anche applicazioni di realtà virtuale, che vedono l'uso di Kinect per controllare un personaggio che si muove in un ambiente 3D, che viene visualizzato tramite un set di occhiali visori.

2.1.1 Campi di utilizzo

Il 29 dicembre 2010 è stato annunciato che il celeberrimo videogioco fantasy tridimensionale, di tipo MMORPG, della *Blizzard Entertainment World of Warcraft* sarà perfettamente giocabile tramite il sistema di riconoscimento gestuale di Kinect. L'effetto viene ottenuto emulando, in tempo reale e senza alcun lag[1], i comandi da tastiera di WoW[2] tramite i comandi gestuali di Kinect.

Kinect è uno strumento nato come componente aggiuntivo per la consolle XBOX 360, quindi

[1] Lag (abbreviazione di *Latency Gap*) è un termine inglese utilizzato nel campo dell'informatica e significa *ritardo*.

[2] World of Warcraft

il contesto principale rimane quello dei videogiochi. Alcuni esempi in commercio che utilizzano Kinect come unico controller sono *Kinect Adventures*, *Kinect Animals* ed il gioco di ballo *Dance Central*.
Il costo relativamente basso insieme alle funizionalità di *body-tracking* che il dispositivo offre, ha fatto smuovere ed incuriosire la massa di sviluppatori software. Dopo qualche mese infatti il Web si è popolato di una moltitudine di applicazioni non strettamente legate al contesto dei videogames. Tra queste si possono citare programmi di visualizzazione di immagini, di riconoscimento del volto, plugin per software già esistenti e addirittura prototipi di riproduzione di una persona attrverso l'utilizzo di due Kinect.

Attualmente è in studio un nuovo sistema di gestione della periferica, che permette di portare il tempo di lavorazione delle immagini da 150 a 5 ms arrivando ad ottenere 200 elaborazioni circa al secondo, in modo da annullare la lag.

2.1.2 Impatto commerciale

La NPD Group, società specializzata in inchieste di mercato, ha rivelato che Microsoft ha venduto due milioni e cinquecentomila unità Kinect nei primi venticinque giorni di commercializzazione.

Al CES (*Consumer Electronics Show*) di Las Vegas, Steve Balmer, amministratore delegato di Microsoft, ha confermato che al 31 dicembre 2010 risultano vendute 8 milioni di unit Kinect in tutto il mondo.
Microsoft contava di vendere un totale di cinque milioni di unità entro il 2011, pertanto le previsioni, da molti considerate esageratamente ottimistiche, sono state, al contrario, troppo conservative.

Il Kinect si conferma un eccezionale dispositivo nonché un eccezionale fenomeno di mercato, gudagnandosi il primato nel *Guinness World Records* per il maggior numero di dispositivi di elettronica di consumo venduti.

2.2 Hardware

Kinect è un dispositivo dotato di telecamera RGB, doppio sensore di profondità a raggi infrarossi composto da un proiettore a infrarossi e da una telecamera sensibile alla stessa banda. La telecamera RGB ha una risoluzione di 640 480 pixel, mentre quella a infrarossi usa una matrice di 320 240 pixel.

Kinect dispone anche di un array di microfoni utilizzato dal sistema per la calibrazione dell'ambiente in cui ci si trova. Mediante l'analisi della riflessione del suono sulle pareti e sull'arredamento infatti, per riconoscere correttamente i comandi vocali, elimina i rumori di fondo e il riverbero causato dai suoni del gioco.

La barra del Kinect è motorizzata lungo l'asse verticale e segue i movimenti dei giocatori, orientandosi nella posizione migliore per il riconoscimento dei movimenti.

Di fatto, la periferica permette all'utente di interagire con la console senza l'uso di alcun controller da impugnare, ma solo attraverso i movimenti del corpo, i comandi vocali o attraverso gli oggetti presenti nell'ambiente. La tecnologia di Kinect riesce a codificare le informazioni che le servono nel momento stesso in cui la luce viaggia, analizzando le deformazioni incontrate nel suo percorso.
Quello che ne deriva è una vera e propria renderizzazione 3D dell'ambiente in tempo reale, molto più precisa che in passato. Le eventuali persone presenti vengono poi rilevate attraverso un chip, piazzato proprio all'interno della videocamera, il quale va alla ricerca di qualsiasi cosa sembri di natura umana e ne calcola i relativi movimenti.

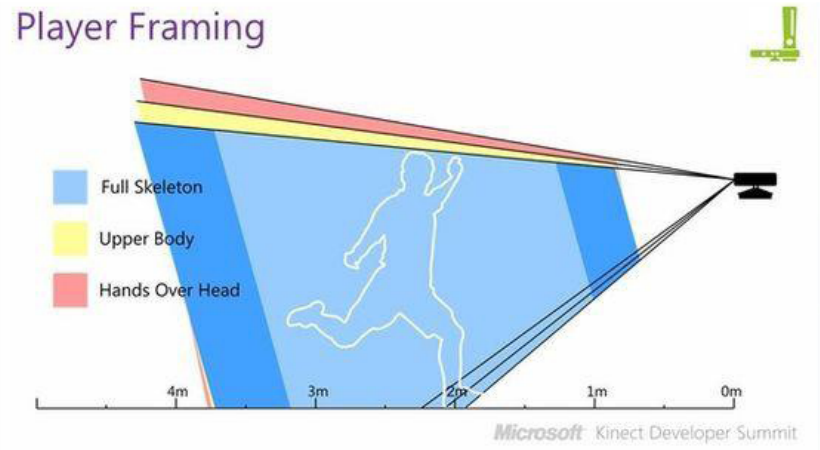


Figura 2.2: *Raffigurazione della profondità di gioco.*

Sensore:

- Lenti sensibili al colore e alla profondità;
- Microfono;
- Tilt motor per permettere alla periferica di spostarsi;
- Compatibile con tutte le console Xbox 360;

Campo visivo:

- Orizzontale: 57 Gradi;
- Verticale: 43 Gradi;

- Capacità di spostamento della periferica: 27 gradi;
- Profondità: 1.2m - 3.5m;

Trasferimento dati:

- 320240 16-bit profondità @ 30 frames/sec;
- 640480 32-bit colore @ 30 frames/sec;
- 16-bit audio @ 16 kHz;

Tracking dello Scheletro:

- Fino a 6 persone, inclusi 2 giocatori attivi;
- Fino a 20 movimenti per ogni giocatore attivo;
- Riconoscimento dell'Avatar dell'utente;

Audio:

- Possibilità di effettuare le *Chat Party*;
- Cancellazione dell'eco;
- Riconoscimento vocale di più voci;

2.3 Kinect SDK

Questo paragrafo si focalizza sulla compatibilità tra Microsoft Kinect e i vari sistemi operativi presenti nel mondo dell'informatica. Introduce inoltre i vari metodi di approccio alla programmazione, confrontando principalmente le SDK ufficiali fornite da Microsoft e quelle non ufficiali basate sul *framework* OpenNI.

2.3.1 Installazione cross platform

Microsoft Kinect è un dispositivo che può tranquillamente essere installato in qualsiasi sistema operativo. È possibile consultare la guida che comprende installazione e primo utilizzo di Kinect per i principali sistemi operativi nell'appendice A.
L'applicazione basata sul dispositivo Kinect ampiamente descritta nel prossimo capitolo si basa sull'utilizzo delle SDK[3] ufficiali fornite da Microsoft. È doveroso precisare però che l'installazione dei driver ufficiali permette di utilizzare e sviluppare applicazioni basate su Kinect esclusivamente in un PC con sistema operativo Windows (Xp, Vista, Seven).

[3] Software Development Kit.

Le SDK ufficiali sono state rilasciate da Microsoft nel luglio del 2011, mentre driver non ufficiali che sono usciti molti mesi prima hanno permesso agli sviluppatori di software di testare il funzionamento di Kinect e di creare una moltitudine di applicazioni basate sul dispositivo lanciato da Microsoft nell'inverno 2010.
I driver non ufficiali sono basati sul *framework* OpenNI (Open Natural Interaction). OpenNI è un *framework* multi-lingua, *cross platform* che definisce le API[4] per scrivere applicazioni che sfruttano la *Natural Interaction* (NI). Le API di OpenNI sono composte da un insieme di interfacce per la scrittura di applicazioni NI. Lo scopo principale del *framework* è quello di formare delle API standard che consentono la comunicazione con:

- Sensori audio e video;
- *Middleware* di percezione audio-visiva (componenti software che analizzano e comprendono i dati audio e video che vengono registrati dalla scena). Per esempio, il software che riceve i dati visivi come un'immagine, restituisce la posizione del palmo di una mano rilevati all'interno dell'immagine;

Il *framework* OpenNI è un livello astratto che fornisce l'interfaccia per entrambi i dispositivi fisici e componenti middleware (rappresentazione grafica in figura 2.3). Le API permettono la registrazione di più componenti nel *framework* OpenNI. Questi componenti sono indicati come moduli, e sono utilizzati per produrre ed elaborare i dati sensoriali. La selezione di un dispositivo hardware più adatto ad un determinato scopo, o di un componente middleware diventa facile e flessibile.

Nel dicembre 2010, PrimeSense ha rilasciato i propri driver *open source* con il *middleware* di *motion tracking* chiamato NITE, menzionato ed utilizzato nella guida di installazione in appendice A.
PrimeSense è un'azienda di semiconduttori, che produce tecnologie ad alte prestazioni per macchina basate sulla riproduzione visiva tridimensionale in vendita per fascia media di consumatori.

La combinazione del *framework* OpenNI con il *middleware* NITE permette dunque di utilizzare Microsoft Kinect non solo con il sistema operativo Windows.

2.3.2 OpenNI e Microsoft Kinect SDK

Entrambe i metodi di installazione dei driver per Kinect consentono di utilizzare delle applicazioni d'esempio. Per la precisione l'installazione di OpenNI in combinazione con NITE PrimeSense offre un numero maggiore di applicazioni d'esempio rispetto al pacchetto eseguibile rilasciato da Microsoft.
Le applicazioni più importanti svolgono pressappoco le stesse funzioni. Entrambe tracciano lo scheletro di una persona, visualizzano il video a colori ed il video che rappresenta la profondità

[4]In informatica con il termine API (*Application Programming Interface*) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma.

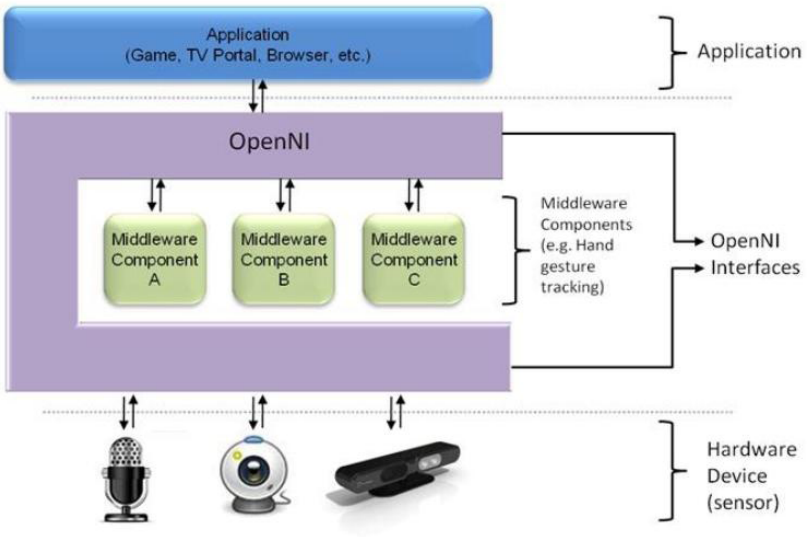


Figura 2.3: *Rappresentazione grafica del framework OpenNI.*

secondo una scala di grigi che l'hardware Kinect fornisce al computer.
L'approccio di programmazione seguito nel capitolo 3 sfrutta la comodità dei prodotti Microsoft. Lavorare con i driver forniti dalla casa madre e utilizzare come ambiente di sviluppo il software Microsoft Visual Studio risulta molto più comodo per programmatori alle prime armi rispetto alla combinazione OpenNI-NITE. Un notevole svantaggio si riscontra nel fatto che le applicazioni create seguendo questo tipo di approccio non sono *cross platform*, ma funzionano esclusivamente con il sistema operativo Windows.
La combinazione OpenNI-NITE si basa sull'utilizzo delle librerie *OpenGL*[5] per visualizzare lo scheletro dell'utente. Il tracciamento dello scheletro avviene tramite una posizione "chiave", ovvero l'utente deve rimanere in posizione eretta, allargare le braccia e posizionare gli avambracci con un angolo di novanta gradi circa rispetto alle braccia. Come si può facilmente dedurre, è alquanto scomodo assumere una particolare posizione per procedere con la lettura dello scheletro, ciò infatti riduce il numero di applicazioni possibili (ad esempio risulta impossibile creare un'applicazione che si basa sul tracciamento dello scheletro di persone affette da evidenti disabilità motorie).

Le applicazioni fornite da Microsoft invece sfruttano le librerie grafiche DirectX[6], utilizzabili esclusivamente in sistemi operativi Windows, ma cosa ben più importante, non è neccesaria alcuna posizione chiave per la rilevazione dello scheletro da parte del dispositivo Kinect. Questo è senza dubbio un enorme vantaggio che Microsoft offre attraverso l'applicazione campione più importante chiamata "*Skeletal Viewer*".

[5]OpenGL (Open Graphics Library) è una specifica che definisce una API per più linguaggi e per più piattaforme per scrivere applicazioni che producono computer grafica 2D e 3D.

[6]DirectX (in origine chiamato "*Game SDK*") è una collezione di API per lo sviluppo semplificato di videogiochi per Windows.

Altra fondamentale caratteristica del pacchetto di installazione ufficiale Microsoft consiste nel poter sfruttare un file di "soluzione" del formato utilizzato da *Microsoft Visual Studio*.
Questo semplifica notevolmente la realizzazione del progetto descritto nel capitolo 3, vista la scelta di utilizzare il software *Microsoft Visual Studio* come IDE[7].

[7](Integrated Development Environment) ambiente per lo sviluppo di applicazioni.

Capitolo 3

Sviluppo del progetto

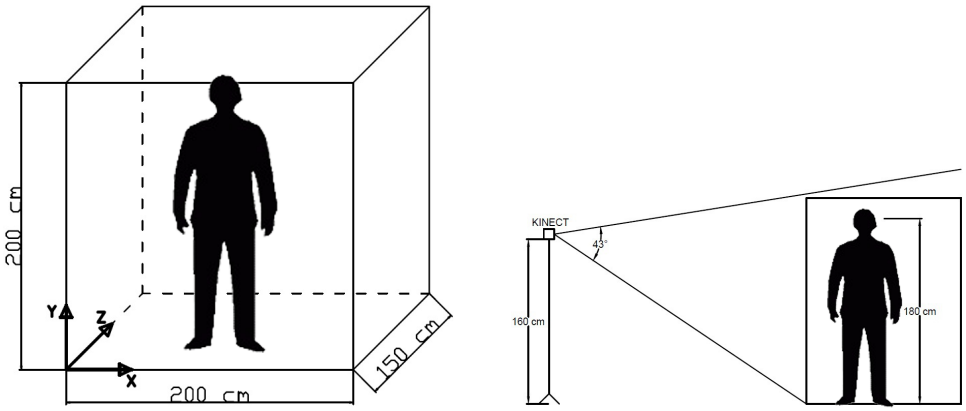
Il progetto mira a studiare le potenzialità del dispositivo Microsoft Kinect come strumento di acquisizione dati legati alla lettura dei movimenti umani.
L'applicazione *Localization in Controlled Space* (LCS) è stata realizzata utilizzando il linguaggio di programmazione C++ modificando ed inserendo nuovi metodi alla soluzione dell'applicazione *Skeletal Viewer* (SV), descritta nella sezione 3.3.3.
I riferimenti a tutte le modifiche apportate al codice del programma originale SV e i metodi creati per la realizzazione del progetto sono descritti nelle sezioni dell'appendice B

3.1 Dimensionamento della stanza controllata

Il dimensionamento della stanza controllata ha lo scopo di calcolare lo spazio più grande all'interno del quale il dispositivo Kinect traccia per intero lo csheletro di una persona.
Il punto di partenza della raccolta dati è accedere ai campi di visualizzazione delle coordinate di Kinect, inviare i valori di output nell'area della finestra principale dedicata e procedere con la lettura dei valori utilizzando il metodo di raccolta dei dati descritto nel prosieguo del paragrafo.

Secondo le informazioni raccolte da internet (`http://support.xbox.com/it-it/pages/kinect/more-topics/sensor-placement.aspx`), l'altezza ideale di posizionamento del dispositivo varia dai 60 centimetri ai 180 centimetri.
Per la raccolta dei dati e per la costruzione della stanza virtuale, l'altezza scelta per il dispositivo è di 160 centimetri circa. L'applicazione avrà inoltre in dotazione un metodo per il controllo del motore interno al dispositivo Kinect per modificare l'angolazione della visuale ed effettuare una calibrazione manuale (B.7).
Decisa un'altezza, non rimane che verificare lo spazio più grande nel quale viene tracciato l'intero scheletro di un utente. Dopo un'attenta analisi sperimentale e con riferimento alla scheda tecnica del paragrafo 2.2, lo spazio maggiore nel quale il dispositivo legge l'intero scheletro dell'utente equivale ad una stanza di larghezza circa 200 centimetri, 200 centimetri di altezza e soli 150 centimetri di profondità.

Si ottiene così un parallelepipedo virtuale di dimensioni 200 x 200 x 150 [cm], entro il quale viene tracciato lo scheletro completo dalla testa ai piedi di una persona e viene effettuata la lettura delle coordinate del punto che rappresenta la testa.
Il passo successivo consiste nel rappresentare fisicamente l'area di base della stanza, in modo da avere un riferimento visivo tangibile e preciso sulla posizione dell'utente all'interno dello spazio controllato. È stato dunque scelto di delimitare l'area del pavimento con del nastro, creando un riferimento visibile per riscontrare la veridicità delle coordinate una volta convertite e adeguate al sistema di riferimento con origine nel vertice inferiore sinistro del parallelepipedo (figura 3.1).



(a) Stanza virtuale controllata (b) Posizionamento Kinect

Figura 3.1: *Rappresentazione dello spazio virtuale di controllo.*

Eseguite tutte le operazioni appena descritte non rimane che procedere con la lettura dei dati per via sperimentale. Il primo passo consiste nell'effettuare le letture delle coordinate di profondità. Sempre con l'ausilio del nastro, vengono suddivisi i 150 centimetri di profondità ad intervalli regolari di 50 centimetri per ottenere un riferimento ancor più preciso.
I valori relativi ad ogni posizione, sono riportati nella tabella 3.1[1].

Si vedrà come nel paragrafo 3.2 le due funzioni di cambiamento di coordinate per la coordinata x e la coordinata y saranno funzioni a due variabili d'ingresso.
Per la coordinata x sono state effettuate le misure lungo i quattro vertici della stanza. Le coordinate fanno riferimento alla testa di una persona di 190 centimetri di altezza (tabella 3.2).

I valori riportati nelle tabelle sono stati distinti tra i valori forniti dal dispositivo Kinect (denominati con la lettera "K") ed i valori reali misurati in centimetri (denominati con la lettera "R").

[1]La tabella 3.1 le ultime due cifre dei valori sotto la voce z^K [ushort] variano talmente velocemente da rendere impossibile una lettura completa, per tanto non sono state prese in considerazione e quindi sostituite con il valore zero.

Punti	z^R [cm]	z^K [ushort]
D_1	0	179200
	50	219400
	100	255200
D_2	150	280000

Tabella 3.1: *Dati profondità*

I punti D_1 e D_2 della tabella 3.1 e i punti delle seguenti tabelle che possiedono un nome saranno utilizzati per il calcolo delle funzioni di cambiamento di coordinate nel paragrafo successivo.

In modo del tutto analogo, vengono effettuate le letture della posizione all'interno dell'intera area di base della stanza virtuale, con la consapevolezza che le coordinate x e y del punto della testa adattate al sistema di riferimento scelto, non dipendono esclusivamente ai valori che il dispositivo Kinect assegna a tali coordinate, ma dipendono anche dalla profondità. Questo avviene perché Kinect fornisce le coordinate del punto proiettato nello schermo.

Punti	x^R [cm]	x^K [pixel]	z^R [cm]
H_1	0	403	0
	100	258	0
H_2	200	110	0
	0	378	50
	100	257	50
	200	133	50
	0	365	100
	100	257	100
	200	146	100
H_3	0	361	150
	100	257	150
	200	157	150

Tabella 3.2: *Dati della coordinata x dipendenti dalla profondità*

Per le misure della coordinata y della tabella 3.3, i rilevamenti sono stati effettuati con un approccio simile al precedente. Sono state rilevate le coordinate riferite all'altezza di una persona di 190 centimetri a profondità zero centimetri e a profondità 150 centimetri. Analogamente, sono stati effettuati i rilevamenti sulla medesima persona, ma in questo caso la posizione non era eretta ma inginocchiata e sollevata dai talloni.

Va precisato che il punto associato alla testa non rappresenta la totale altezza dell'utente, dal-

l'esempio SV infatti si nota come esso sia posizionato all'incirca a metà della testa, pertanto l'altezza associata al rilevamento dell'altezza di una persona di 190 centimetri viene ridotta a 180 centimetri, circa dieci in meno dell'altezza reale.

Punti	y^R [cm]	y^K [pixel]	z^R [cm]
V_1	180	357	0
	180	324	50
	180	292	100
	180	267	150
V_2	130	377	0
V_3	130	283	150

Tabella 3.3: *Dati della coordinata y dipendenti dalla profondità*

Tutti i rilevamenti sono stati effettuati con una precisa inclinazione (denominata ϕ) del sensore Kinect.
Ogni qualvolta che l'applicazione LCS viene aperta, il dispositivo viene automaticamente settato all'angolo ϕ ($\phi \simeq -13°$).

3.2 Il sistema di riferimento

Come già detto, l'applicazione LCS fornisce in uscita le coordinate della testain grandezze intere con precisione del centimetro, pertanto da un punto di vista puramente informatico risulta più semplice lavorare con dati di tipo intero (*integer*).
Va ripetuto che Kinect fornisce dei dati associati alla profondità molto precisi, mentre le coordinate x e y descrivono la posizione del punto proiettata nello schermo misurata in pixel, pertanto le funzioni che convertono tali coordinate sono delle funzioni a due variabili che dipendono anche dalla profondità.
Per comodità si definiscono le dimensioni della stanza come segue:

$$[0, \mathbf{w}],\ [0, \mathbf{h}],\ [0, \mathbf{d}];$$

con le costanti di *width*, *heigth* e *depth* assumenti i valori (valore espresso in centimetri):

$$\mathbf{w} = 200,\ \mathbf{h} = 200,\ \mathbf{d} = 150;$$

La funzione di cambiamento delle coordinate della profondità è la 3.2 ed è basata sul segmento interno alla stanza appartenente alla retta ricavata dai punti D_1 e D_2 di equazione 3.1 esplicitata per la variabile z^R. La funzione fa riferimento al primo dei tre metodi scritti in linguaggio C++ riportati nella sezione B.1.

La funzione è semplicemente l'equazione di una retta ed è stata ricavata dai due punti D_1 e D_2 della tabella 3.1.

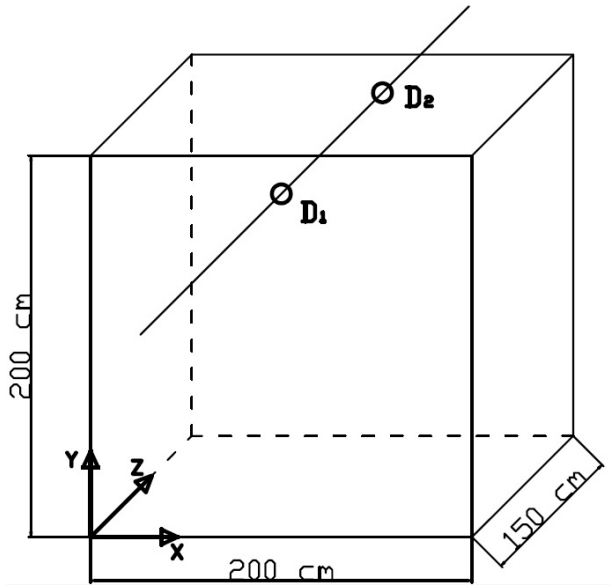


Figura 3.2: *Retta che attraversa i punti D_1 e D_2.*

$$f(z^K) = z^R = a \cdot z^K - b \tag{3.1}$$

$$g(z^R) = \begin{cases} z^R & \text{se} \quad 0 \leq z^R \leq \mathbf{d} \\ 0 & \text{altrove} \end{cases} \tag{3.2}$$

con le costanti che valgono:

$$a = \frac{z^R_{D_2}}{z^K_{D_2} - z^K_{D_1}},\ b = \frac{z^K_{D_1} \cdot z^R_{D_2}}{z^K_{D_2} - z^K_{D_1}};$$

A differenza della z^R, le variabili x^R e y^R dipendono anche dalla profondità per i motivi precedentemente evidenziati. Quest'ultime richiedono un calcolo leggermente più complesso, poiché la funzione di cambiamento di coordinate è una funzione a due variabili. La funzione che restituisce il valore in centimetri della coordinata x è la 3.4 e rappresenta la porzione del piano orizzontale di equazione 3.3 (ricavata utilizzando i punti H_1, H_2 e H_3 della tabella 3.2), interna alla stanza virtuale controllata.

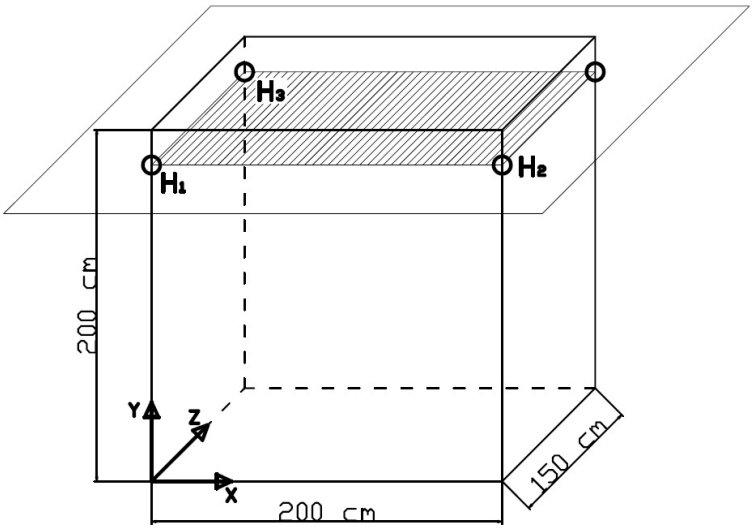


Figura 3.3: *Piano che attraversa i punti H_1, H_2 e H_3.*

Equazione semplificata di un piano nello spazio:

$$a \cdot x^R + b \cdot x^K + c \cdot z^R + d = 0;$$

da cui il sistema di equazioni parametriche di un piano nello spazio:

$$\begin{cases} x^R = \lambda_1 u + \lambda_2 v + \alpha \\ x^K = \mu_1 u + \mu_2 v + \beta \\ z^R = \nu_1 u + \nu_2 v + \gamma \end{cases}$$

da cui:

$$\begin{cases} x^R = (x^R_{H_2} - x^R_{H_1})u + (x^R_{H_3} - x^R_{H_1})v + x^R_{H_1} \\ x^K = (x^K_{H_2} - x^K_{H_1})u + (x^K_{H_3} - x^K_{H_1})v + x^K_{H_1} \\ z^R = (z^R_{H_2} - z^R_{H_1})u + (z^R_{H_3} - z^R_{H_1})v + z^R_{H_1} \end{cases}$$

quindi:

$$\lambda_1 = x^R_{H_2} - x^R_{H_1} = -\frac{b}{a},\ \lambda_2 = x^R_{H_3} - x^R_{H_1} = -\frac{c}{a},\ \alpha = x^K_{H_1} = -\frac{d}{a};$$

Tornando all'equazione semplificata di un piano nello spazio, si esplicita tale equazione per la x^R:

$$x^R = -\frac{b}{a} \cdot x^K - \frac{c}{a} \cdot z^R - \frac{d}{a}$$

Che sostituendo diventa:

$$f(x^K, z^R) = x^R = (x^R_{H_2} - x^R_{H_1}) \cdot x^K - (x^R_{H_3} - x^R_{H_1}) \cdot z^R - x^K_{H_1} \tag{3.3}$$

infine risulta la funzione:

$$g(x^R) = \begin{cases} x^R & \text{se} \quad 0 \leq x^R \leq \mathbf{w} \quad \text{e} \quad 0 \leq z^R \leq \mathbf{d} \\ 0 & \text{altrove} \end{cases} \tag{3.4}$$

Il procedimento matematico di calcolo della funzione che determina il valore della coordinata y^R è del tutto analogo al precedente. La funzione 3.5 dunque, rappresenta l'equazione di un piano ricavata dai punti V_1, V_2 e V_3 appartenenti alla tabella 3.3.

$$f(y^K, z^R) = y^R = -\frac{b}{a} \cdot y^K - \frac{c}{a} \cdot z^R - \frac{d}{a}; \tag{3.5}$$

con:

$$g(y^R) = \begin{cases} y^R & \text{se} \quad 0 \leq y^R \leq \mathbf{h} \quad \text{e} \quad 0 \leq z^R \leq \mathbf{d} \\ 0 & \text{altrove} \end{cases} \tag{3.6}$$

con le costanti che valgono:

$$y^R_{H_2} - y^R_{H_1} = -\frac{b}{a},\ y^R_{H_3} - y^R_{H_1} = -\frac{c}{a},\ y^K_{H_1} = -\frac{d}{a};$$

È doveroso precisare che tutti e tre i metodi di cambiamento delle coordinate appartenenti al programma LCS, sfruttano altrettanti metodi che verificano se le coordinate (x^R, y^R e z^R) sono interne o meno alla stanza virtuale di dimensioni 200 x 200 x 150 [cm]. Queste funzioni statiche molto semplici sono descritte nel listato B.2 nella relativa appendice.

3.3 L'applicazione

All'interno di questo paragrafo viene esposto il procedimento di creazione dell'applicazione *Localizzation in Controlled Space* a partire dall'applicazione *Skeletal Viewer* (SV). Viene descritto

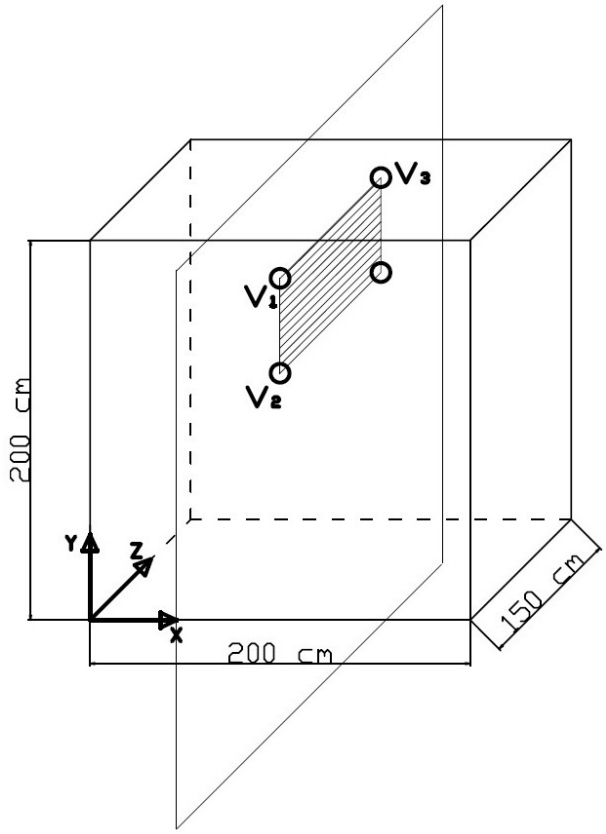


Figura 3.4: *Piano che attraversa i punti V_1, V_2 e V_3.*

il funzionamento di SV ed i metodi principali che la compongono, le modifiche grafiche apportate a SV per procedere con la creazione dell'interfaccia grafica di LCS ed infine viene spiegato come vengono disegnati il punto relativo alla testa ed il punto di sorgente sonora.

3.3.1 L'esempio Skeletal Viewer

L'installazione delle SDK Microsoft (versione Beta 1 installate il mese di agosto 2011), come precedentemente esposto, comprende due esempi di applicazioni per il dispositivo Kinect, con tanto di file soluzione del formato adatto al programma *Microsoft Visual Studio*, un ambiente di sviluppo integrato chiamato più comunemente IDE. L'esempio più interessante è chiamato *Skeletal Viewer* (figura 3.5), l'unico programmato con linguaggio di programmazione C++.
Tale applicazione è composta da una finestra che contiene tre riquadri di visualizzazione, uno per la vista di profondità con scala di grigi ed il riconoscimento degli utenti (player), un riquadro per visualizzare lo scheletro di uno o al massimo due utenti identificati con il sensore di profondità e un riquadro per visualizzare l'immagine a colori della telecamera. Nel lato destro invece, viene visualizzato il numero di fotogrammi al secondo (frame rate), ossia la frequenza di cattura dei fotogrammi che compongono il filmato.

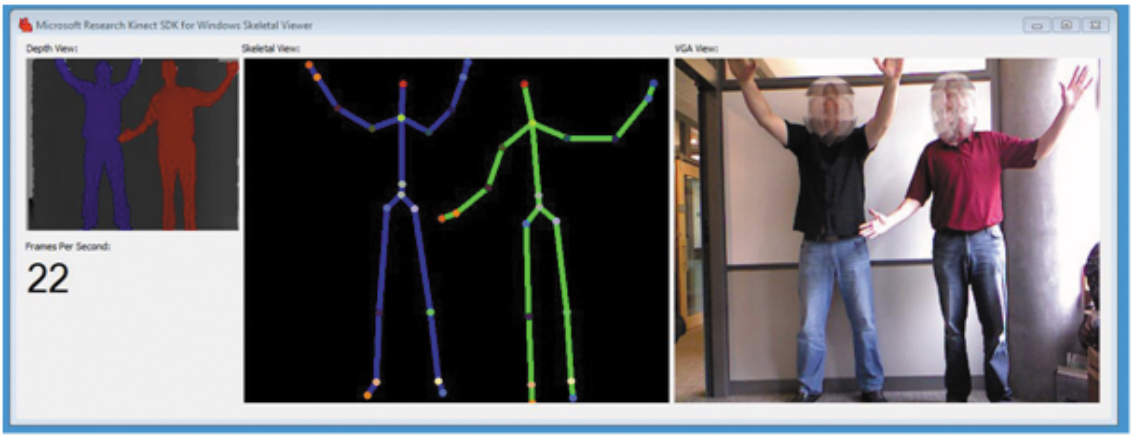


Figura 3.5: *Applicazione SkeletalViewer.*

L'applicazione *Skeletal Viewer* funziona in maniera molto semplice. Per prima cosa è necessario che ci sia una o più persone presenti all'interno dell'area visualizzata dalla telecamera (riquadro *VGA View*), dopodiché l'applicazione riconosce automaticamente gli utenti e colora l'area di ciascuno di essi utilizzando colori differenti e casuali all'interno del riquadro associato alla profondità (riquadro *Depth View*).

Dopo aver riconosciuto almeno una persona, l'applicazione ne traccia lo scheletro all'interno del secondo riquadro (riquadro *Skeletal View*), tale scheletro è composto da dei punti che corrispondono alle giunture più significative del corpo umano, tali punti sono uniti da segmenti e lo scheletro segue praticamente in tempo reale i movimenti del corpo umano dell'utente (movimenti troppo veloci posso causare ritardi).
Sostanzialmente questo software ha come funzionalità di base il riconoscimento di uno o più utenti ed il conseguente tracciamento dello scheletro di massimo due persone.

Skeletal Viewer dispone di molti metodi all'interno del file *NuiImpl.cpp*. Prima di procedere con la modifica della soluzione del programma per realizzare l'applicazione LCS è necessario capire il funzionamento dei principali metodi:

- **Metodo *Nui_Init***: crea tre eventi, il primo per l'immagine video VGA, il secondo associato all'immagine video con scala di grigi per la profondità ed il terzo per la creazione dello scheletro. Crea i *buffer*[2] video per riprodurre le immagini delle camere di Kinect nei riquadri *VGA view* e *depth view*, imposta la risoluzione dei video e apre gli *stream*[3]. Inizializza inoltre l'ambiente grafico per il disegno dello scheletro all'interno del riquadro *skeletal view*;

[2]Un buffer è una zona di memoria usata temporaneamente per l'input o l'output dei dati oppure per velocizzare l'esecuzione di alcune operazioni.

[3]Uno stream (anche detto flusso) è un canale tra la sorgente e la destinazione attraverso il quale fluiscono i dati.

- **Metodo *Nui_ProcessThread***: gestisce gli eventi e comunica direttamente con la *main window*. Invia e aggiorna continuamente i valori che popolano i campi dell'interfaccia grafica garantendo una visualizzazione dei dati in tempo reale. Grazie alla gestione degli eventi, il metodo *Nui_ProcessThread*, effettua delle chiamate ad altri metodi nei momenti più appropriati, ad esempio quando si verifica un evento "scheletro", viene chiamato il metodo *Nui_DrawSkeleton* che disegna lo scheletro di un utente nel riquadro dedicato della finestra principale;

- **Metodo *Nui_DrawSkeletonSegment***: disegna i segmenti che compongono lo scheletro. Questo metodo viene chiamato all'interno del metodo *Nui_DrawSkeleton* e viene utilizzato per collegare una serie di punti (ad esempio i segmenti che vanno da una mano all'altra, passando per braccia e spalle);

- **Metodo *Nui_DrawSkeleton***: rileva e posiziona tutti i venti punti che compongono la rappresentazione dello scheletro dell'utente. Chiama il metodo *Nui_DrawSkeletonSegment* per collegare i punti con dei segmenti. È all'interno di questo metodo che vengono estratti i valori della posizione del punto riferito alla testa utilizzati dalle funzioni di cambiamento delle coordinate del paragrafo 3.2.

3.3.2 Modifica dell'interfaccia grafica

Un passaggio fondamentale per la realizzazione dell'applicazione LCS utilizzando l'applicazione SV come punto di partenza, consiste nel modificare l'interfaccia grafica del programma inserendo i riquadri nei quali visualizzare la proiezione dall'alto e la proiezione frontale della stanza controllata (come si vedrà in seguito), i campi contenenti le coordinate del punto relativo alla testa e al punto di sorgente sonora e inserendo del testo statico utile per specificare unità di misura, nomi delle variabili e testo di semplice spiegazione.
Il tracciamento della testa è fondamentale per utilizzare LSC nel campo delle applicazioni che sfruttano l'audio binaurale, come prefisso nel capitolo 1.
È possibile apprezzare uno *screenshot* dell'applicazione in fase di costruzione in figura 3.6.

Tutte le righe di codice aggiunte durante la programmazione sono riportate nella sezione B.8. Sostanzialmente vengono aggiunti degli oggetti grafici nei quali va specificato nome, dimensioni e se hanno un comportamento statico o dinamico.
Confrontando la figura 3.5 relativa all'applicazione SV con la figura 3.6, si possono constatare le numerose aggiunte apportate in fase di programmazione.

Esclusi i riquadri ereditati dal precedente programma, già descritti nel precedente paragrafo, si noti infatti come lungo il lato sinistro dell'applicazione siano presenti i campi utilizzati per visualizzare le coordinate spaziali dei due punti. Tutte le misure raccolte per procedere con il metodo sperimentale descritto nel paragrafo 3.1 sono state effettuate visualizzando in questi campi il valore delle coordinate che Kinect fornisce.
Nella parte più bassa della finestra invece sono stati creati due riquadri nei quali viene visualizza-

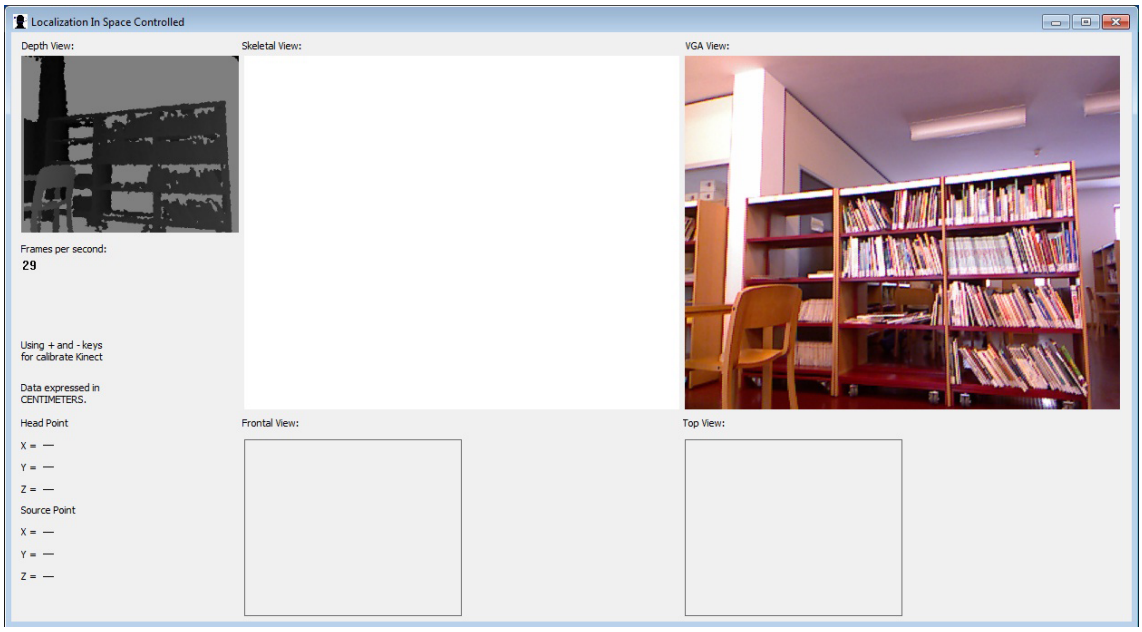


Figura 3.6: *Modifiche grafiche apportate all'applicazione Skeletal Viewer.*

ta la proiezione frontale e la proiezione dall'alto della stanza virtuale controllata, in un secondo momento all'interno di queste proiezioni viene monitorata la posizione del punto relativo alla testa (punto rosso) e del punto di sorgente sonora (punto blu) visibili nell'immagine 3.7 (di cui i dettagli nel paragrafo successivo) che da un'idea del funzionamento dell'applicazione finale.

Ogni campo è dotato di adeguate stringhe di testo statico fondamentali per specificare nome ed unità di misura. Stringhe di testo statico sono state utilizzate anche per fornire informazioni riguardo al funzionamento dell'applicazione (sotto il riquadro *depth view*), per specificare i tasti chiave con i quali il punto di sorgente sonora viene spostato e per specificare il nome di ogni riquadro.

Come precisazione conclusiva si sottolinea come il dispositivo Kinect debba avere un'angolazione precisa per il corretto funzionamento dell'applicazione, pertanto ogni volta che l'applicazione viene aperta, il dispositivo si posiziona ad una angolazione predefinita. Con il tasto "-" ed il tasto "+" invece viene variato l'angolo del dispositivo Kinect per effettuare una eventuale calibrazione.

3.3.3 Head-point e sorgente sonora

Nei due riquadri dedicati alle proiezioni, come operazione preliminare sono disegnati i perimetri delle prospettive della stanza con i relativi assi di riferimento, all'interno dei quali il moto del punto riferito alla testa si muove in base alla posizione della persona all'interno dello spazio vir-

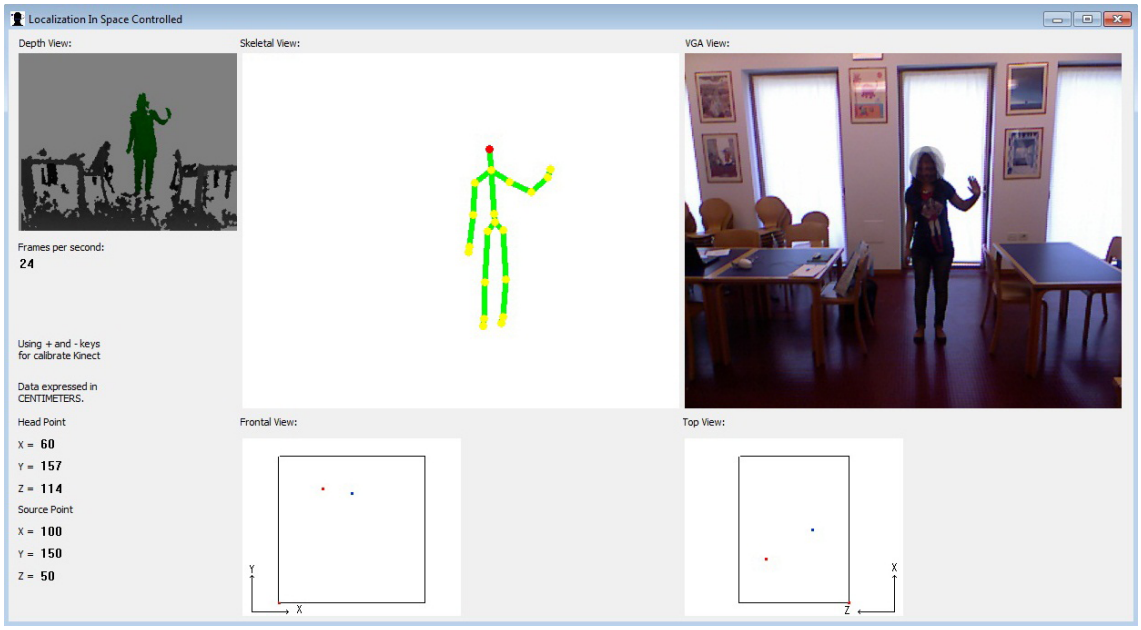


Figura 3.7: *Applicazione Localization in Controlled Space.*

tuale controllato.
I metodi utilizzati per questo scopo anche se sono composti da molte righe di codice, sono molto semplici e sono riportati nella sezione B.3. Tali metodi, se l'area del riquadro ancora non è stata inizializzata, colorano lo sfondo (bianco nel nostro caso), dopodiché vengono semplicemente create delle linee che vanno a disegnare il perimetro, i due assi cartesiani orientati e i nomi degli assi.

A differenza dei metodi che disegnano le proiezioni della stanza, i metodi che disegnano i punti (sezione B.4) devono essere continuamente chiamati all'interno di *Nui_ProcessThread*, per aggiornare continuamente la posizione dell'utente.
Un vettore (*array* in linguaggio informatico) fornisce i valori della reale posizione dei punti espressa in centimetri. Per un soggetto che utilizza l'applicazione LCS, i dati espressi con l'unità di misura dei centimetri sono molto utili in quanto agevola la lettura e l'interpretazione, ma il centimetro non è un'unità di misura adatta per essere utilizzata dai metodi che disegnano i punti a seconda delle coordinate.
Per disegnare correttamente la posizione dei punti all'interno dello spazio controllato e fare in modo che essi rappresentino l'effettiva posizione della testa e della sorgente sonora è fondamentale scalare adeguatamente le coordinate espresse in centimetri.
Ciò avviene attraverso quattro metodi statici creati appositamente per questo scopo. I metodi, riportati nella sezione B.6, sono molto semplici e hanno lo scopo di adattare le coordinate del punto, in modo da poter realizzare una rappresentazione bidimensionale dei punti all'interno delle proiezioni.

È fondamentale precisare che le coordinate del punto che rappresenta la posizione della testa, dal momento che sono prelevate all'interno del metodo *Nui_DrawSkeleton*, variano in continuazione poiché dipendono dal movimento dell'utente.
La funzione che disegna il secondo punto (punto di sorgente sonora), è del tutto analoga alla funzione che disegna il punto della testa. Le coordinate riferite alle due proiezioni della stanza virtuale che la funzione ottinene in ingresso, sono adeguatamente scalate grazie le funzioni descritte precedentemente.

L'applicazione LCS risponde alla pressione di alcuni tasti della tastiera, ogni tasto chiama una funzione descritta all'interno del file *nuiInit.cpp*. Con i tasti freccia della tastiera viene spostato il punto di sorgente sonora nella vista dall'alto. Grazie ai tasti *Pag Up/Down* invece viene modificata l'altezza del punto, tale variazione si può apprezzare nella prospettiva frontale della stanza. Questi spostamenti avvengono attraverso metodi modificatori (codice della sezione B.5), che hanno lo scopo di modificare le coordinate del punto di sorgente sonora, permettendone lo spostamento all'interno dello spazio controllato.

Capitolo 4

Conclusioni e sviluppi futuri

L'applicazione LCS descritta nel precedente capitolo:

- traccia lo scheletro di un utente posizionato davanti al dispositivo Kinect, monitora il punto relativo alla posizione della testa,
- definisce il setup grazie al quale si ottiene lo spazio maggiore all'interno del quale viene disegnato l'intero scheletro,
- visualizza la stanza controllata attraverso le due prospettive *frontal view* e *top view*, tracciandovi l'*head-point*,
- crea e modifica la posizione di un punto di sorgente sonora all'interno dello spazio virtuale,
- fornisce le coordinate di questi punti in base ad un sistema di riferimento creato ad hoc.

Questa applicazione è un primo passo verso la realizzazione di ambienti audio 3D utilizzando lo strumento Kinect. Grazie al tracciamento della testa vengono fornite informazioni necessarie per il rendering spazializzato di una sorgente sonora, alla base di applicazioni che utilizzano l'audio binaurale (descritte nel primo capitolo).

4.1 Analisi e problematiche

È da sottolineare che il dispositivo Kinect non è esente da imprecisioni, le quali si riflettono nell'applicazione LCS. Non è un fattore trascurabile che il dispositivo fornisca le proiezioni sullo schermo dei punti misurate in pixel. Questo si riflette in una forte imprecisione nel rilevamento dell'altezza dopo la conversione pixel/centimetri. È possibile notare questa imprecisione nella proiezione frontale, nella quale l'*head-point*, per l'elevata escursione all'interno del cono visivo di Kinect, e perché è il punto più alto dello scheletro, compie delle oscillazioni lungo l'asse verticale non indifferenti.
L'impostazione sperimentale utilizzata per lo sviluppo del progetto non valuta la posizione del punto che rappresenta la testa è molto elevato rispetto all'orizzonte di Kinect.

L'applicazione LCS potrebbe sfruttare un protocollo per comunicare con dispositivi multimediali e non, diventando quindi una componente importante nella creazione di un ambiente audio 3D.
Per fare ciò si utilizza *Open Sound Control* (OSC) è un protocollo per la comunicazione tra computer, sintetizzatori audio e altri dispositivi multimediali, ottimizzato per la tecnologia di networking moderna. I vantaggi comprendono interoperabilità, precisione, flessibilità e maggiore organizzazione e documentazione.

4.2 Proposte di miglioramenti

Durante lo svluppo del progetto inerente all'applicazione LCS, è stato adottato un approccio sperimentale per determinare lo spazio più grande all'interno del quale Kinect rileva l'intero scheletro dell'utente. Una possibile alternativa potrebbe consistere nello sfruttare i dati tecnici riportati nel paragrafo 2.2 per calcolare lo spazio controllato attraverso un'approccio analitico.

Il sensore Kinect fornisce le coordinate dei punti del corpo ed identifica con precisione la profondità nella quale sono presenti una o più persone. Le applicazioni binaurali (trattate nel capitolo 1) necessitano di una renderizzazione audio 3D tramite cuffie oppure tramite diffusori sonori posizionati lungo il perimetro di una stanza. In quest'ultimo caso, ci si rende conto dell'importanza non solo dell'*head-tracking*, ma anche della direzione verso cui volge lo sguardo di un utente.
Dato che Kinect non rileva queste caratteristiche, se non a distanza ravvicinata, si potrebbero utilizzare dei sensori giroscopici interfacciati con il dispositivo. Un sensore giroscopico è un dispositivo fisico rotante che tende a mantenere il suo asse di rotazione orientato in una direzione fissa, questa potrebbe essere una possibile soluzione al problema dell'orientamento della testa.

Il volume della stanza virtuale controllata calcolato nel precedente capitolo ha delle dimensioni piuttosto contenute. L'ampliamento dello spazio di lettura dello scheletro di un utente è un aspetto che merita di essere migliorato. Si potrebbe pensare all'utilizzo di due o più dispositivi Kinect, creando un programma che permette di interpretare i dati ricevuti da ciascun dispositivo, migliorando o aumentando lo spazio di lettura.

Appendici

Appendice A

Guida all'installazione dei driver per Microsoft Kinect

A.1 Installazione dei driver ufficiali (Windows)

Scaricare ed installare la versione beta delle SDK per Microsoft Kinect direttamente dal sito Kinect for Windows SDK from Microsoft Research, sotto la sezione *download*.
Eseguire semplicemente il file **KinectSDK32.exe** per installare driver e le applicazioni.

A.2 Installazione dei driver non ufficiali (Windows)

L'installazione del *framework* OpenNI e di NITE PrimeSense in Windows (Xp, Vista, Seven) è molto semplice.
Per prima cosa è necessario installare i driver per Microsoft Kinect nel PC, lo si può fare direttamente dal sito avin2/SensorKinect - GitHub.
Una volta scaricato e scompattato il pacchetto, navigare all'indirizzo Platform\Win32\Driver ed eseguire il file **dpinst-x86.exe** (per processori a 32bit) oppure il file **dpinst-amd64.exe** (per processori a 64bit), infine eseguire il file **sensorKinect-Win32-5.0.0.exe**.

Dopo questa prima operazione, i driver saranno installati nel PC. A questo punto, scaricare ed installare la versione stabile o instabile degli OpenNI *Binaries file* dal sito OpenNI - OpenNI Home.
In modo del tutto analogo, si procede con linstallazione del *middleware* PrimeSense, utilizzando durante linstallazione la key: **0KOIk2JeIBYClPWVnMoRKn5cdY4=** non appena verrà richiesta.
Scaricare ed installare i file OpenNI *Compilant Hardware Binaries* (da scegliere sempre tra la versione stabile o instabile).

A questo punto collegare Kinect alla porta USB del PC, ed aspettare che la periferica hardware venga riconosciuta.
Ora non rimane che scaricare i file *KinectXMLs* (KinectXMLs.zip) per poter utilizzare le prime

applicazioni campione di OpenNI e di PrimeSense.
Scaricati e scompattati questi file XML già compilati, entrare nella directory KinectXMLs\OpenNI, copiare il contenuto ed incollarlo, sostituendo i file presenti, in C:\Programmi\ OpenNI\Data.
Ripetere tale operazione per poter utilizzare le applicazioni del Middleware PrimeSense. Copiare il contenuto della cartella KinectXMLs\NITE ed incollarlo in C:\Programmi\Prime Sense\ NITE\Data sostituendo i file esistenti.
Conclusi tutti i passaggi non rimane che provare le applicazioni campione di OpenNI (C:\Programmi\OpenNI\Samples\Bin\Release)
e di NITE (C:\Program Files\Prime Sense\NITE\Samples\Bin\Release).

Il dispositivo Microsoft Kinect dispone anche di un motore posto alla base che ne controlla la posizione. Il *framework* OpenNI e tutti i pacchetti installati non permettono tale controllo.
Per poter utilizzare tutte le risorse di Kinect c'è bisogno dellinstallazione del programma che funge da accelerometro, chiamato "*CL NUI Device Test*". Lo si può scaricare direttamente dal sito CodeLaboratories Downloads scaricando il pacchetto chiamato *CL NUI Platform*.

A.3 Installazione dei driver non ufficiali (Linux)

La guida all'installazione che segue è riferita alla distribuzione più utilizzata del sistema operativo Linux, ovvero Ubuntu (versione 10.04).

È possibile installare i driver non ufficiali basati sulle librerie del *framework* OpenNI, insieme al *middleware* NITE PrimeSense nelle versioni di Ubuntu superiori alla 10.10.

La guida più generale e completa per installare i driver non ufficiali in Ubuntu 10.04 è presente nel sito Getting Started/it - OpenKinect. A tal proposito è necessario anticipare che le applicazioni che si andranno ad installare (glview e glpcview) sono molto differenti dalle applicazioni fornite da OpenNI e NITE.
Per procedere con l'installazione è necessario specificare che in Linux è necessaria un po' più di manualità, non vengono installati i file eseguibili classici come avviene nei sistemi operativi Windows, ma in ogni caso nulla di estremamente complicato da richiedere la competenza di un utente esperto.
Come prima operazione è necessario scaricare i pacchetti per libfreenect dal sito , dopodichè le operazioni sono piuttosto semplici e consistono semplicemente di copiare ed incollare nel terminale le seguenti righe di comando. Per usare il ppa (*Personal Package Archives*) *launchpad* per Ubuntu, eseguire il comando da terminale:

sudo add-apt-repository ppa:arne-alamut/freenect

Dopo di che si deve sincronizzare nuovamente l'elenco dei pacchetti dalle *repositorty*:

sudo apt-get update

Poi sempre da terminale eseguire:

sudo apt-get install freenect

Con questo comando verranno installati i pacchetti *libfreenect0.0*, *libfreenect-demo* e *libfreenect-dev*.
Fatto questo è necessario aggiungere se stessi come utenti al gruppo "video" ed effettuare nuovamente il login. Il pacchetto include già le regole necessarie per *udev* in modo che l'accesso al dispositivo sia possibile per gli utenti nel gruppo "video".

sudo adduser TUONOME video

A questo punto basta collegare Kinect tramite la presa USB (estrarre ed introdurre la presa USB nel caso il dispositivo fosse già collegato in precedenza).
Infine avviare le applicazioni demo digitando:

freenect-glview Il nome della demo è glview, in generale per avviare le demo disponibili il comando è *freenect-NOMEDEMO*.

A.4 Installazione dei driver non ufficiali (OS X)

Prima di procedere con l'installazione del *framework* OpenNI e del *middleware* NITE PrimeSense, per poi utilizzare Microsoft Kinect in un Mac sono necessari alcuni prerequisiti come i programmi MacPorts, Fink oppure Homebrew per facilitare l'installazione di software open source.
Si scelga ad esempio MacPorts, ad installazione avvenuta eseguire i comandi:

sudo port install git-core

sudo port install libtool

sudo port install libusb-devel

Poi spostarsi nella directory di lavoro ed eseguire il comando:

git clone https://github.com/OpenKinect/libfreenect.git

Clonare le repository OpenKinect e Github:

git clone https://github.com/OpenKinect/libfreenect.git

git clone git://git.libusb.org/libusb.git

Configurare OpenKinect:

cd ../libfreenect/

mkdir build

cd build

ccmake ..

È possibile che compaia il messaggio "*Empty cache*" all'avvio. Premete "c" in *ccmake* per configurarlo. Se *libusb* è stato installato tramite MacPorts funzionerà senza ulteriori modifiche. In caso contrario, è probabile un esito negativo, perché *libusb* non verrà trovata. Premete "e" per uscire dall'help e modificate manualmente il *path libusb* nella schermata successiva, in modo che punti a /usr/local/include/libusb-1.0/ e continuare.
Eseguite tutte le operazioni, compilare OpenKinect:

cmake ..

make

Al termine di queste operazioni dovrebbe essere presente un programma chiamato *glview* in *libfreenect/build/bin*. A questo punto non rimane che collegare Kinect e avviare *glview*. Per avere i file disponibili a livello globale sul vostro Mac e utilizzarli con i vostri progetti, li potete installare:

sudo make install

Conclusa l'installazione

Appendice B

Codice dei metodi principali dell'applicazione *Localizzation In Space Controlled*

B.1 Cambiamento delle coordinate

Listing B.1: Cambiamento coordinate

```
int zToCent(int value)
{
        const double m = 150/10100;
        int z_depth =   static_cast<int>(value * m);
        z_depth = z_depth - 266;
        if(zRange(z_depth))
                return z_depth;
        else
                return 0;
}
int xToCent(int x_value, int z_value)
{
        double bra = 0.682594;
        double cra = 0.191126;
        double dra = 275.085324;

        int xCent = static_cast<int>(dra - (bra * x_value) - (cra * z_value));

        if(xRange(xCent) && zRange(z_value))
                return xCent;
        else
                return 0;
}
int yToCent(int y_value, int z_value)   //      MOLTO SENSIBILE
{
        double bra = 2.647058;
        //double cra = 1.658823;
        double dra = 1127.941176;

        double cra = 1.641176;

        int yCent = static_cast<int>(dra - (bra * y_value) - (cra * z_value));
```

```
        if(yRange(yCent) && zRange(z_value))
                return yCent;
        else
                return 0;
}
```

B.2 Verifica posizione dell'Head-Point

Listing B.2: Verifica range

```
BOOLEAN xRange(int width)
{
        if((0 <= width) && (width <= 200))
                return TRUE;
        else
                return FALSE;
}
BOOLEAN yRange(int height)
{
        if((0 <= height) && (height <= 200))
                return TRUE;
        else
                return FALSE;
}
BOOLEAN zRange(int depth)
{
        if((0 <= depth) && (depth <= 150))
                return TRUE;
        else
                return FALSE;
}
```

B.3 Disegno delle proiezioni

Listing B.3: Disegno delle proiezioni della stanza virtuale

```
void CSkeletalViewerApp::Nui_DrawFrontalRoom( bool bBlank, HWND hWnd)
 {
         HDC hdc = GetDC( hWnd );
    RECT rct;
    GetClientRect(hWnd, &rct);
    int width = rct.right;
    int height = rct.bottom;

        if(bBlank)
        {
                PatBlt( hdc, 0, 0, width, height, WHITENESS );
        }

        HPEN hJointPen;
        hJointPen = CreatePen(PS_SOLID,2, g_MyJointColorTable[1]);
        HGDIOBJ hOldObj = SelectObject(hdc,hJointPen);
```

```
        MoveToEx( hdc, 40, 20, NULL );
        LineTo( hdc, 200, 20);
        LineTo( hdc, 200, 180);
        LineTo( hdc, 40, 180);
        LineTo( hdc, 40, 20);            //      CREO IL RETTANGOLO

        SelectObject( hdc, hOldObj );
    DeleteObject(hJointPen);

        HPEN hJointPen2;
        hJointPen2 = CreatePen(PS_SOLID,1, g_MyJointColorTable[1]);
        HGDIOBJ hOldObj2 = SelectObject(hdc,hJointPen2);

        MoveToEx( hdc, 10, 190, NULL );
        LineTo( hdc, 50, 190);
        MoveToEx( hdc, 10, 190, NULL );
        LineTo( hdc, 10, 150);
        MoveToEx( hdc, 50, 190, NULL );
        LineTo( hdc, 47, 187);
        MoveToEx( hdc, 50, 190, NULL );
        LineTo( hdc, 47, 193);
        MoveToEx( hdc, 10, 150, NULL );
        LineTo( hdc, 7, 153);
        MoveToEx( hdc, 10, 150, NULL );
        LineTo( hdc, 13, 153);                  //      ASSI CARTESIANI

        MoveToEx( hdc, 8, 139, NULL );
        LineTo( hdc, 10, 143);
        LineTo( hdc, 10, 148);
        MoveToEx( hdc, 12, 139, NULL );
        LineTo( hdc, 10, 143);                  //      LETTERA Y


        MoveToEx( hdc, 60, 191, NULL );
        LineTo( hdc, 64, 182);
        MoveToEx( hdc, 64, 191, NULL );
        LineTo( hdc, 60, 182);                  //      LETTERA X

        SelectObject( hdc, hOldObj2 );
   DeleteObject(hJointPen2);

        return;
}// END METODO
void CSkeletalViewerApp::Nui_DrawTopRoom( bool bBlank, HWND hWnd)
 {
         HDC hdc = GetDC( hWnd );
    RECT rct;
    GetClientRect(hWnd, &rct);
    int width = rct.right;
    int height = rct.bottom;

        if(bBlank)
        {
                PatBlt( hdc, 0, 0, width, height, WHITENESS );
        }
        HPEN hJointPen;
        hJointPen = CreatePen(PS_SOLID,2, g_MyJointColorTable[1]);
        HGDIOBJ hOldObj = SelectObject(hdc,hJointPen);

        MoveToEx( hdc, 60, 20, NULL );
        LineTo( hdc, 180, 20);
        LineTo( hdc, 180, 180);
        LineTo( hdc, 60, 180);
        LineTo( hdc, 60, 20);            //      CREO IL RETTANGOLO
```

```
        SelectObject( hdc, hOldObj );
    DeleteObject(hJointPen);

        HPEN hJointPen2;
        hJointPen2 = CreatePen(PS_SOLID,1, g_MyJointColorTable[1]);
        HGDIOBJ hOldObj2 = SelectObject(hdc,hJointPen2);

        MoveToEx( hdc, 230, 190, NULL );
        LineTo( hdc, 190, 190);
        MoveToEx( hdc, 230, 190, NULL );
        LineTo( hdc, 230, 150);
        MoveToEx( hdc, 190, 190, NULL );
        LineTo( hdc, 193, 187);
        MoveToEx( hdc, 190, 190, NULL );
        LineTo( hdc, 193, 193);
        MoveToEx( hdc, 230, 150, NULL );
        LineTo( hdc, 227, 153);
        MoveToEx( hdc, 230, 150, NULL );
        LineTo( hdc, 233, 153);                    //       ASSI CARTESIANI

        MoveToEx( hdc, 228, 145, NULL );
        LineTo( hdc, 232, 136);
        MoveToEx( hdc, 228, 137, NULL );
        LineTo( hdc, 232, 146);                    //       LETTERA X

        MoveToEx( hdc, 180, 192, NULL );
        LineTo( hdc, 176, 192);
        LineTo( hdc, 180, 184);
        LineTo( hdc, 175, 184);                    //       LETTERA Z

        SelectObject( hdc, hOldObj2 );
    DeleteObject(hJointPen2);

        return;
}// END METODO
```

B.4 Disegno dei punti

Listing B.4: Disegno dei punti

```
void CSkeletalViewerApp::Nui_DrawPoint( bool bBlank, HWND hWnd, int xCoord, int yCoord)
 {
        HDC hdc = GetDC( hWnd );
    RECT rct;
    GetClientRect(hWnd, &rct);
    int width = rct.right;
    int height = rct.bottom;

        if(bBlank)
        {
                PatBlt( hdc, 0, 0, width, height, WHITENESS );
        }

        HPEN hJointPen;
        hJointPen = CreatePen(PS_SOLID,4, g_MyJointColorTable[0]);
        HGDIOBJ hOldObj = SelectObject(hdc,hJointPen);

        MoveToEx( hdc, xCoord, yCoord, NULL );
        LineTo( hdc, xCoord, yCoord);
```

```
        SelectObject( hdc, hOldObj );
    DeleteObject(hJointPen);

        ReleaseDC(hWnd,hdc);

        return;
}// END

void CSkeletalViewerApp::Nui_DrawHeadPoint( bool bBlank, HWND hWnd, int xCoord,
int yCoord, bool front_up)
 {
        HDC hdc = GetDC( hWnd );
    RECT rct;
    GetClientRect(hWnd, &rct);
    int width = rct.right;
    int height = rct.bottom;

        if(front_up)
                PatBlt( hdc, 40, 20, 160, 160, WHITENESS );
        else
                PatBlt( hdc, 60, 20, 120, 160, WHITENESS );

        HPEN hJointPen;
        hJointPen = CreatePen(PS_SOLID,4, g_MyJointColorTable[2]);
        HGDIOBJ hOldObj = SelectObject(hdc,hJointPen);

        MoveToEx(hdc, xCoord, yCoord, NULL );
        LineTo(hdc, xCoord, yCoord);

        SelectObject(hdc, hOldObj );
    DeleteObject(hJointPen);

        ReleaseDC(hWnd,hdc);

        return;
}// END
```

B.5 Spostamento del punto di sorgente sonora

Listing B.5: Spostamento del punto di sorgente sonora nello spazio

```
void CSkeletalViewerApp::Nui_MoveUpSourcePoint()
{
        int shift = source[0] + 3;
        if(xRange(shift))
                source[0] = shift;
}
void CSkeletalViewerApp::Nui_MoveDownSourcePoint()
{
        int shift = source[0] - 3;
        if(xRange(shift))
                source[0] = shift;
}
void CSkeletalViewerApp::Nui_MoveRightSourcePoint()
{
        int shift = source[2] - 3;
        if(zRange(shift))
                source[2] = shift;
}
void CSkeletalViewerApp::Nui_MoveLeftSourcePoint()
```

```
{
        int shift = source[2] + 3;
        if(zRange(shift))
                source[2] = shift;
}
void CSkeletalViewerApp::Nui_MoveUpYSourcePoint()
{
        int shift = source[1] + 3;
        if(yRange(shift))
                source[1] = shift;
}
void CSkeletalViewerApp::Nui_MoveDownYSourcePoint()
{
        int shift = source[1] - 3;
        if(yRange(shift))
                source[1] = shift;
}
```

B.6 Adattamento coordinate

Listing B.6: Adattamento delle coordinate alle proiezioni della stanza

```
int xAdaptFrontal(int x_position)
{
        int xf = int (0.8*x_position + 40);
        return xf;
}
int yAdaptFrontal(int y_position)
{
        int yf = int (0.8*y_position);
        yf = 180 - yf;
        return yf;
}

//      UP ROOM SCALE FUNCTION
int zAdaptTop(int z_position)
{
        int zu = int (0.8*z_position);
        zu = 180 - zu;
        return zu;
}
int xAdaptTop(int x_position)
{
        int xu = int (0.8*x_position);
        xu = 180 - xu;
        return xu;
}
```

B.7 Modifica dell'angolo di posizione del dispositivo

Listing B.7: Spostamento angolare

```
void CSkeletalViewerApp::Nui_MoveUpCamera()
{
```

```
        LONG delta = cameraAngle + 1;
        if(delta <= 30)
                cameraAngle = delta;
}
void CSkeletalViewerApp::Nui_MoveDownCamera()
{
        LONG delta = cameraAngle - 1;
        if(delta >= -30)
                cameraAngle = delta;
}

void CSkeletalViewerApp::Nui_SetAngle(LONG angle)
{
        NuiCameraElevationSetAngle(angle);
}
```

B.8 Modifica dell'interfaccia grafica

Listing B.8: Modifica dell'interfaccia grafica

```
CONTROL         "",IDC_MY_ROOM_FRONTAL,"Static",SS_BLACKFRAME,171,276,160,120
CONTROL         "",IDC_MY_ROOM_TOP,"Static",SS_BLACKFRAME,495,276,160,120
LTEXT           "Head Point Coordinates:",IDC_STATIC,7,261,68,8
LTEXT           "X = ",IDC_STATIC,7,276,12,8
LTEXT           "---",IDC_XH,23,275,21,8
LTEXT           "Y = ",IDC_STATIC,7,291,12,8
LTEXT           "---",IDC_YH,23,290,21,8
LTEXT           "Z = ",IDC_STATIC,7,306,12,8
LTEXT           "---",IDC_ZH,23,305,21,8
LTEXT           "Source Point Coordinates:",IDC_STATIC,7,320,68,8
LTEXT           "X = ",IDC_STATIC,7,335,12,8
LTEXT           "---",IDC_XP,23,334,21,8
LTEXT           "Y = ",IDC_STATIC,7,350,12,8
LTEXT           "---",IDC_YP,23,349,21,8
LTEXT           "Z = ",IDC_STATIC,7,365,12,8
LTEXT           "---",IDC_ZP,23,364,21,8
LTEXT           "Frontal View:",IDC_STATIC,170,261,48,8
LTEXT           "Top View:",IDC_STATIC,494,261,48,8
LTEXT           "Data expressed in CENTIMETERS.",IDC_STATIC,7,237,68,16
LTEXT           "Using + and - keys for calibrate Kinect device.",IDC_STATIC,7,208,68,16
```

B.9 Utilizzo dei tasti chiave della tastiera

Listing B.9: Chiamata dei metodi modificatori

```
LRESULT CALLBACK CSkeletalViewerApp::DialogControlProc
(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        switch(message)
        {
                case WM_SETFOCUS:
                        {
                                return DefWindowProc(hWnd, message, wParam, lParam);
                        }
```

```
                case WM_GETDLGCODE:
                        {
                                return DLGC_WANTARROWS;
                        }
                case WM_KEYDOWN:
                        {
                                switch(wParam)
                                {
                                case VK_DOWN:
                                        ScheletroApp.Nui_MoveDownSourcePoint();
                                        break;
                                case VK_UP:
                                        ScheletroApp.Nui_MoveUpSourcePoint();
                                        break;
                                case VK_RIGHT:
                                        ScheletroApp.Nui_MoveRightSourcePoint();
                                        break;
                                case VK_LEFT:
                                        ScheletroApp.Nui_MoveLeftSourcePoint();
                                        break;

                                case VK_PRIOR:
                                        ScheletroApp.Nui_MoveUpYSourcePoint();
                                        break;
                                case VK_NEXT:
                                        ScheletroApp.Nui_MoveDownYSourcePoint();
                                        break;

                                case VK_OEM_PLUS:
                                        ScheletroApp.Nui_MoveUpCamera();
                                        break;
                                case VK_OEM_MINUS:
                                        ScheletroApp.Nui_MoveDownCamera();
                                        break;
                                }//end switch
                                return 0;
                        }//end WM_KEYDOWN
        }
        return DefDlgProc(hWnd, message, wParam, lParam);
}
```

Bibliografia

[1] E. M. Wenzel e M. R. Anderson D. R. Begault. Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. *Journal of the Audio Engineering Society*, pages 904–916, 2001.

[2] D. J. Kistler e F. L. Wightman E. M. Wenzel, M. Arruda. Localization using nonindividualized head-related transfer functions. *The Journal of the Acoustical Society of America*, pages 111–123, 1993.

[3] F. Avanzini M. Geronazzo, S. Spagnol. Customized 3d sound for innovative interaction design. *Non specificato*, pages 1–3, 2011.

[4] A. Utsumi e F. Kishino P. Milgram, H. Takemura. Augmented reality: A class of displays on the reality-virtuality continuum. *SIAM Rev.*, pages 282–292, 1994.

[5] Benjamin Woolley. *Mondi Virtuali*. Bollati Boringhieri, Torino, 1993.