TVR:

UN VISUALIZZATORE VOLUMETRICO AD ALTE PRESTAZIONI BASATO SU OpenGL 1.0

Antonio Zorcolo, Piero Pili, Enrico Gobbetti

CRS4

Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna
Via Nazario Sauro 10, 09123 Cagliari

{Antonio.Zorcolo|Piero.Pili|Enrico|Gobbetti}@crs4.it

1. Sommario

Ritenendo che uno strumento di questo tipo sia di grande utilità nella pianificazione degli interventi in campo medicale, si è realizzato un sistema di visualizzazione stereoscopica di dati medici volumetrici con tracciamento della posizione dell'osservatore, che consente sia la possibilità di trovare nel modo più naturale (un semplice spostamento del capo) la posizione più idonea per osservare la patologia che la piena comprensione della struttura tridimensionale della patologia stessa. L'utilità dello strumento è accresciuta dalla possibilità, offerta dalla collimazione dello spazio virtuale del visualizzatore con quello fisico dell'osservatore, di manipolare direttamente il volume virtuale utilizzando le mani come sul volume reale.

In questo rapporto viene descritta la struttura del visualizzatore, denominato TVR, e analizzate in dettaglio le principali scelte implementative. I primi capitoli presentano i pesanti vincoli a quali s'è dovuto sottostare mentre i capitoli successivi illustrano le soluzioni adottate per soddisfarli. Negli ultimi capitoli sono riportate infine le prestazioni raggiunte e gli obiettivi futuri.

2. Obiettivi

Uno dei motivi principali per cui i medici fanno uso di dispositivi di acquisizione e visualizzazione quali CT, MRI, PET ecc. è quello di individuare le caratteristiche di una patologia (natura, dimensioni ecc.) con l'intento di pianificare la più idonea strategia d'intervento. A tal fine, essi hanno per lungo tempo utilizzato le immagini bidimensionali relative alle varie sezioni acquisite. Questo metodo forza i medici a combinare delle immagini separate per formarsi un modello mentale tridimensionale degli oggetti, la cui complessità comporta spesso un ulteriore appesantimento di un processo già intrinsecamente difficile.

La creazione di una ambiente virtuale per l'analisi di dati volumetrici supera questi problemi imponendo una serie di requisiti alla visualizzazione. I più importanti sono il supporto alla visione stereoscopica e alla parallassi di movimento:

- *La visione binoculare*:
  Una coppia di immagini della stessa scena è in questo caso acquisita simultaneamente dagli occhi dell'osservatore. La ricostruzione tridimensionale può avvalersi di informazioni ulteriori quali la distanza di messa a fuoco, la convergenza delle direzioni di vista e la distanza interoculare.

- *La parallassi di movimento*:
  Consente la valutazione delle distanze grazie all'aquisizione non contemporanea di differenti viste dell'oggetto attraverso semplici spostamenti del punto di vista (movimenti della testa dell'osservatore o dell'oggetto rispetto all'osservatore).
  Rispetto al caso della semplice osservazione di due immagini statiche (ad es. due fotografie) l'osservatore può in questo caso disporre di una sequenza di immagini conseguente al un suo spostamento continuo e progressivo intorno all'oggetto. L'interpretazione delle immagini è in questo caso facilitata anche dalla loro correlazione con lo spostamento eseguito.
  Grazie alla parallassi di movimento è possibile ricostruire correttamente la sagoma di un oggetto anche mantenendo chiuso un occhio.

Figura 1: Tracciamento della posizione dell'osservatore durante la visualizzazione stereoscopica di un volume ricostruito da dati CT.

L'obiettivo principale che ci si è proposti durante la realizzazione del visualizzatore è proprio quello di consentire al medico (o a qualunque altro potenziale utente del visualizzatore) *una corretta e naturale percezione di strutture tridimensionali anche complesse* grazie all'ausilio della stereoscopia e del motion parallax.

Nel perseguire questo obiettivo si è inoltre posta particolare attenzione a non trascurare un terzo naturale canale di percezione della forma degli oggetti: la risposta alla illuminazione. Sebbene questo aspetto non sia strettamente legato alle altre due caratteristiche del TVR, costituisce però un ulteriore importante apporto alla comprensione della scena.

Un secondo importante obiettivo che ci si è proposti di raggiungere col visualizzatore TVR è quello di fornire all'utente *modalità d'interazione col volume il più possibile naturali*.

In un normale visualizzatore 2D (o comunque un visualizzatore che presenta scene tridimenesionali proiettate sullo schermo bidimensionale) l'utente si trova a lavorare in uno spazio che non è il proprio e ad eseguire uno sforzo per *immergersi* nello spazio dell'applicazione. Tale sforzo è dovuto non solo alla visione di una proiezione bidimensionale dell'oggetto ma anche alla necessità di manipolarlo utilizzando gli 'utensili' bidimensionali resi disponibili dall'applicazione stessa.

Nel caso del TVR, non solo l'oggetto è presentato nello spazio fisico dell'osservatore (esso appare sospeso davanti allo schermo cosicchè l'osservatore è istintivamente portato a 'toccarlo' con le mani) ma è anche offerta la possibilità di manipolarlo direttamente utilizzando utensili virtuali tridimensionali nello stesso modo in cui si utilizzerebbero utensili reali.

Tutto ciò è reso possibile dalla esatta sovrapposizione dello spazio virtuale con lo spazio fisico.

Figura 2: Manipolazione diretta di piani di taglio.

Appare subito evidente come le potenzialità del TVR non sono limitate solamente ad una realistica percezione dell'oggetto visualizzato, ma si estendono a tutti i possibili campi di applicazione in cui si richiede di simulare sull'oggetto una azione che possa comprometterne l'integrità. L'asportazione di aree di tessuto degradate, l'inserimento di innesti o protesi, l'introduzione di sonde sono esempi di interventi virtuali che possono consentire di valutare in anteprima l'impatto che gli stessi avrebbero nella realtà sul paziente (o, in altri campi, sulla struttura esaminata).

3. Vincoli e limitazioni

3.1 Sovrapposizione degli spazi

Per sovrapposizione degli spazi si intende la esatta corrispondenza delle coordinate virtuali a quelle fisiche. Ciò è necessario a garantire che tanto la posizione quanto la dimensione degli oggetti valutata dall'osservatore coincidano con quelle valutate dal sistema e viceversa. In altri termini se l'osservatore tocca con un dito la superficie dell'oggetto virtuale, la posizione del dito rilevata dal sistema di tracking deve essere la stessa avvertita dall'osservatore, cosicchè il punto della superficie dell'oggetto valutato nello spazio virtuale coincide esattamente con quello 'toccato' fisicamente dall'osservatore.

Se da un punto di vista puramente geometrico ciò corrisponde all'esecuzione opportuna di semplici trasformazioni di rototraslazione e scalatura (la correttezza di queste trasformazioni implica anche quella della percezione delle distanze per motion parallax), dal punto di vista della visione stereoscopica comporta invece la necessità d'eseguire una procedura di calibrazione atta a determinare l'esatta posizione dei vertici dei volumi di vista rispetto alla testa dell'osservatore. Ciò è dovuto al fatto che la posizione fisica degli occhi dell'osservatore rispetto alla sua posizione tracciata dal sistema è variabile da utente a utente.

La procedura, che deve essere eseguita come prima fase della sessione, è estremamente semplice per l'utente e consiste nel richiedere allo stesso di andare a toccare dei punti di collimazione virtuali noti al sistema che può così valutare lo scarto esistente fra la loro posizione effettiva e quella percepita dall'osservatore. I punti possono essere, ad esempio, i quattro angoli dello schermo e l'azione di toccarli può essere eseguita mediante l'ausilio dei una mouse tridimensionale (tracciato dallo stesso sistema utilizzato per la posizione dell'osservatore) posizionandone il puntatore fisico su tali punti e premendone uno dei bottoni a contatto avvenuto.

3.2 Frequenza fotogrammetrica e tempo di latenza

Due importanti vincoli ai quali il sistema deve sottostare affinchè l'interattività sia effettiva sono la limitazione del tempo di generazione dell'immagine e del tempo di latenza.

Il primo determina il numero di "fotogrammi" che per unità di tempo il sistema è in grado di generare. Tale numero deve essere sufficientemente elevato se si vuole ottenere una certa continuità nel movimento. Frequenze di fotogramma di 10 fps sono il minimo richiesto al di sotto del quale il movimento appare sensibilmente discontinuo[(1)].

Il secondo influenza fortemente la capacità dell'utente di associare ad una sua azione sull'oggetto virtuale il suo effetto sullo stesso.

La soglia di latenza che può essere accettata dipende fortemente dal campo di applicazione al quale il visualizzatore è destinato. Se impiegato per l'addestramento di personale ad attività nelle quali sia importante il tempo di reazione la latenza deve essere contenuta entro poche decine di ms mentre può essere accettato un tempo di latenza di 100 ms per altre applicazioni interattive. In ogni caso latenze superiori a 300 ms comportano per l'utente la dissociazione fra azione ed effetto[2].

3.3 Portabilità

La portabilità del visualizzatore costituisce un ulteriore vincolo che si è ritenuto di dover aggiungere a quelli già menzionati al fine di soddisfare l'esigenza, non trascurabile, di poterlo utilizzare anche su workstations appartenenti ad una fascia di prezzo media o medio-bassa. Sebbene tale scelta può apparire in contrasto con le stesse esigenze prestazionali che ne possano consentire un efficace impiego, la rapidità con la quale cresce oggigiorno il rapporto prestazioni/prezzo dei mezzi di calcolo rende auspicabile che in un prossimo futuro si possa disporre anche su macchine di fascia media o medio-bassa di harware grafico del livello delle migliori macchine attuali.

Tenendo conto di questi vincoli, abbiamo realizzato il visualizzatore TVR in maniera da renderlo compatibile con OpenGL 1.0.

4. Generalità sulle tecniche di rendering

Vengono qui indicate sommariamente le tecniche utilizzate per la realizzazione del TVR che hanno consentito di soddisfare i vincoli precedentemente esposti. I dettagli implementativi sono invece descritti più diffusamente nei capitoli seguenti.

4.1 Visualizzazione diretta

La *Direct Volume Visualization* è la tecnica che consente di presentare contemporaneamente in una sola immagine tutte le sezioni del volume prodotte dal sistema di acquisizione senza passare attraverso rappresentazioni intermedie. Qualunque sia la tecnica successivamente adottata per il rendering, con la visualizzazione diretta le varie sezioni prodotte dal sistema di acquisizione sono impilate nella posizione originaria in modo da ricostruire integralmente il volume. Il particolare algoritmo di rendering si occupa successivamente di *attraversare* il volume per determinare cosa dello stesso sia effettivamente visibile all'osservatore e di proiettarlo sul piano di vista.

4.2 Texture mapping rendering[3][4]

Il requisito principale al quale si è dovuto fare riferimento nella scelta della tecnica di rendering è quello del tempo di esecuzione. Per soddisfare i pesanti vincoli citati in precedenza si è dovuto cercare il massimo sfruttamento dell'hardware grafico messo a disposizione dai più avanzati e moderni acceleratori grafici. Si è dunque optato per la

tecnica del texture mapping con la quale si è potuta confinare la processazione software del volume alla sola fase iniziale e a qualche fase intermedia di peso modesto. Il rendering vero e proprio è invece completamente affidato all'hardware grafico il cui sfruttamento è facilitato dalla predisposizione del volume nel formato più idoneo e dall'adozione di opportune tabelle di look-up.

Il texture mapping si basa sul fatto che ciascuna slice del volume, anche se effettivamente dotata di un certo spessore, può essere trattata come una texture bidimensionale in cui ciascun texel contiene i valori associati al voxel che gli corrisponde (stessa intensità oppure stesse componenti di colore R, G, B e stessa opacità α).

Similmente ad altre tecniche di rendering esegue la fusione back-to-front delle slices del volume ma con la differenza sostanziale che la complessità geometrica della scena gli è decisamente favorevole.

In esso infatti il volume viene ricostruito con una serie di piani paralleli aventi dimensioni pari a quelle del volume stesso (in altezza e larghezza) in numero pari al numero delle slices e a distanza l'uno dall'altro pari allo spessore delle slices; ogni slice diviene semplicemente una decalcomania (la texture) appiccicata al piano che le corrisponde, trasformata in coordinate di vista e proiettata sul piano di vista semplicemente trasformando e proiettando il piano. Si può pensare ai piani come a delle lastre di vetro assolutamente invisibili nell'immagine finale. L'applicazione dell'alpha blending consente di intravedere le textures posteriori attraverso quelle anteriori; l'impressione è quella di vedere l'interno del volume.

Per avere una idea della differenza fra la complessità computazionale del texture mapping e quella di un'altro metodo basato sul merge back-to-front delle slices si prenda in condsiderazione un volume di 128 voxels per lato: anche ammettendo di considerare ogni voxel come un punto dello spazio occupato dal volume (cioè semplicemente come un vertice) 128^3 vertici devono essere trasformati nel metodo del merging contro 128×4 vertici (4 vertici per ciascuno dei 128 piani) trasformati in texture mapping; ci si può attendere dunque che il tempo di esecuzione sia in texture mapping, ammesso che esso sia consumato interamente nel pipeline geometrico, circa 4000 volte inferiore. In realtà il vero collo di bottiglia del texture mapping è proprio il trattamento della texture (sostanzialmente interpolazione delle coordinate e dei valori della texture), per cui le prestazioni effettivamente raggiungibili sono sensibilmente inferiori. Tuttavia l'enorme potenza raggiunta dell'hardware dei più moderni acceleratori grafici è tale che con volumi di dimensioni pari a quelle citate i tempi di generazione dell'immagine possono tranquillamente scendere al di sotto del decimo di secondo.

Si noti che se l'efficienza di un sistema con tali prestazioni fosse tale che il trattamento delle texture richiedesse un tempo dieci volte superiore rispetto a quello complessivamente richiesto dal pipeline geometrico, il tempo richiesto dal metodo del merging per la generazione della stessa immagine sarebbe di circa 40 secondi.

Il texture mapping presenta tuttavia importanti inconvenienti.

Il primo è che la perdita di spessore delle slices comporta che le stesse possano essere osservate solo da posizioni limitatamente angolate rispetto alla loro normale; se l'osservatore si spostasse infatti su un fianco del volume vedrebbe solo il profilo delle slices o meglio dei piani che ospitano le textures.

Il secondo inconveniente è che le figure geometriche che il sistema si trova ad elaborare (i piani di supporto delle textures) non hanno più niente in comune con quelle effettivamente presenti nel volume; il sistema non ha quindi alcuna informazione per determinare come l'oggetto rappresentato risponda alla illuminazione. La conseguenza è che il modello di illuminazione non può essere implementato via hardware se non in modo molto semplificato. Modelli più sofisticati possono essere realizzati via software ma imponendo severe restrizioni che consentano di contenere i tempi di rendering.

Due possibilità utilizzate nel TVR sono:

- L'adozione del distance-only shading che può essere calcolato sui piani di supporto e da

questi trasferito alle textures per blending.

- Il calcolo via software dell'illuminazione dell'oggetto originario come fase di preprocessazione e la successiva creazione di textures già illuminate; in questo modo però l'illuminazione è relativa alla posizione che la sorgente (o le sorgenti) avevano quando il calcolo è stato svolto e non può essere adeguata a successivi riposizionamenti della sorgente, o (cosa che accade molto più frequentemente) a rotazioni del volume, se non ricalcolando le textures.

La libreria grafica, di cui la stazione Silicon Graphics *Infinite Reality* sulla quale il TVR è stato sviluppato è dotata, comprende diverse funzioni espressamente realizzate per la visualizzazione di volumi; esse costituiscono l'estensione SGI per il rendering di volume alla libreria standard OpenGL 1.0. Queste estensioni consentono di utilizzare textures tridimensionali (l'intero volume da visualizzare) dalle quali apposito hardware è in grado di 'ritagliare' textures' bidimensionali in qualunque direzione [5][6][7][8].

Per visualizzare il volume, una volta assegnate le coordinate del punto di vista rispetto al sistema di riferimento del volume stesso, ci si deve preoccupare solamente di definire una serie di settori sferici con centro sul punto di vista (tanti quante sono le slices nelle quali si richiede di suddividere il volume), intersecanti il volume a uguale distanza l'uno dall'altro. Tali settori sono utilizzati dal sistema per determinare le slices, ciascuna delle quali è generata prendendo tutti i punti del volume che cadono sullo stesso settore. Naturalmente non si tratterà di veri e propri settori sferici ma piuttosto di una loro approssimazione mediante poligoni piani; dall'interpolazione delle coordinate 3D dei vertici di ciascun poligono il sistema determina le coordinate dei punti del volume che cadono sul poligono stesso e che identificano i texels componenti la texture.

Il motivo per cui si utilizzano settori sferici piuttosto che semplici piani è quello di far incidere i raggi di vista sempre ortogonalmente alle textures. Ciò garantisce:

- che il ricoprimento delle textures sia massimo per qualunque posizione dell'osservatore;
- che l'opacità della texutre sia la stessa per tutti i punti della texture e per tutte le textures.

Il problema della costanza dell'opacità è estremamente importante. Se si suppone che il volume sia omogeneo, l'opacità complessiva di ciascuna slice per una data direzione di vista dipende dallo spazio che il raggio deve percorrere per attraversarla, cioè dallo spessore della slice e dall'angolo d'incidenza; se le slices hanno tutte lo stesso spessore e sono sempre attraversate ortogonalmente, l'opacità valutata dal sistema sarà sempre la stessa. Ciò non accade invece utilizzando semplicemente dei piani, per i quali si ha il minimo dell'opacità in corrispondenza della perpendicolare alla slice condotta per l'osservatore e il massimo della stessa sui bordi della slices dove, a causa dell'angolazione, è maggiore il percorso che il raggio compie per attraversarla. L'effetto visivo sarebbe quello di una minor trasparenza (o, si preferisce pensare in termini di densità, una maggior densità) del volume nella zona centrale rispetto alle zone periferiche. Se questa disomogeneità può risultare inapprezzabile per immagini statiche (specie se il volume è piccolo in rapporto alla distanza dell'osservatore) è invece decisamente avvertita dall'osservatore durante il movimento relativo dei due; il massimo dell'opacità si sposterebbe nel volume insieme all'osservatore!

Nonostante ciò, il visualizzatore non utilizza le facilitazioni offerte dalle estensioni SGI e visualizza il volume mediante textures bidimensionali piane utilizzando esclusivamente chiamate OpenGL 1.0; questa scelta è stata effettuata per i già citati requisiti di portabilità.

Il risultato è stato però un consistente aumento della complessità del codice necessaria a controllare la corretta generazione dell'immagine in una gamma di condizioni sufficientemente ampia.

Una prima complicazione è stata la necessità di eseguire, per i motivi già esposti, la correzione del valore di opacità della slice in funzione dell'angolo d'incidenza del raggio di vista. Un'altra è stata la necessità di lavorare con tre versioni del volume, ciascuna delle quali aventi le slices ortogonali ad un diverso spigolo dello stesso, per consentirne l'osservazione da qualunque direzione.

5. Il preprocessing

5.1 Lettura dataset e remapping dell'intensità

Il volume generato durante la lettura del dataset non è direttamente utilizzabile per la visualizzazione: le informazioni di cui l'algoritmo deve disporre per il rendering differiscono in generale da quelle originali nel tipo, nella risoluzione e nel formato.

I valori di intensità vengono rimappati su 8 bit. Date le notevoli dimensioni del volume questo consente di ridurre l'occupazione di memoria e/o di sfruttare lo spazio recuperato per associare al valore di intensità altre informazioni utili alla visualizzazione. Il remapping è eseguito direttamente nella fase di caricamento dal dataset se nessuna informazione aggiuntiva deve essere inserita nel volume, altrimenti viene eseguito durante le fasi successive di resampling e di calcolo del gradiente. Eseguendo la classificazione col metodo di Levoy, ad esempio, deve essere inserito nel volume anche il valore del gradiente di intensità per il quale sono necessari almeno altri 4 bit.

5.2 Resampling del volume

Il resampling del volume è una operazione di vitale importanza per il funzionamento dell'algoritmo.

In primo luogo OpenGL richiede che la risoluzione delle textures su entrambi i lati sia una potenza di 2; per alcuni sistemi si ha l'ulteriore restrizione che le textures siano anche quadrate (le estensioni SGI impongono invece textures tridimensionali cubiche). Poichè il dataset non rispetta solitamente nessuna di queste condizioni, il ricampionamento è indispensabile.

In secondo luogo la possibilità di visualizzare il volume indifferentemente da una qualunque delle direzioni principali utilizzando textures bidimensionali comporta la creazione di tre serie di textures, una per ciascuna direzione.

In terzo luogo, anche ammesso che la risoluzione del volume di partenza rispondesse alle limitazioni anzidette, non sarebbe comunque garantito che fosse anche quella ottimale per il raggiungimento del migliore rapporto qualità della immagine / prestazioni.

Infine, seppure si verificasse che il volume originale ha già la risoluzione corretta in tutte le direzioni, sarebbe comunque necessario riorganizzarne la struttura al fine di ottenere la massima velocità di trasferimento delle informazioni dalla RAM al processore e viceversa. Dal momento che la struttura di un array 3D in memoria è comunque unidimensionale, l'accesso più veloce alle informazioni ivi contenute è possibile solo se la sua scansione viene fatta con lo stesso ordine col quale le informazioni si susseguono in memoria. Non solo questo minimizza il peso delle operazioni che il sistema deve svolgere per indicizzare l'array (semplici incrementi unitari se la sequenza è corretta) ma consente sopratutto di ridurre i movimenti fra memoria RAM e memoria cache. Poichè la sequenza corretta può aversi per una sola direzione principale del volume, tre versioni dello stesso, ordinate secondo le tre direzioni principali, devono essere predisposte. Per lo stesso motivo OpenGL richiede che le textures gli siano fornite già ordinate per righe.

Il resampling viene dunque eseguito tre volte, nelle tre direzioni e con la risoluzione più

idonea, avanzando nel volume originario con nX-1, nY-1 e nZ-1 steps nelle rispettive direzioni (dove nX, nY, nZ sono le risoluzioni desiderate) in modo che il primo e l'ultimo campionamento cadano esattamente sul primo e l'ultimo elemento originario per entrambe le direzioni. Per tutti i punti intermedi il campionamento verrà fatto in generale all'interno di una cella del reticolo di partenza per interpolazione trilineare dei valori associati agli 8 vertici.

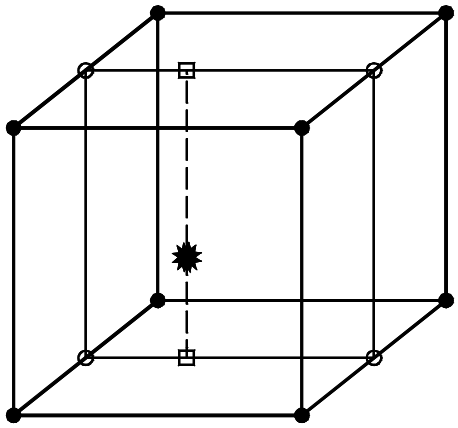

Figura 3: Interpolazione trilineare del volume.

5.3 Calcolo delle normali

La quantità di luce riflessa da una areola della superficie dell'oggetto in un dato punto dipende dalla orientazione della normale nel punto stesso. Poichè fra le informazioni contenute nel dataset non compaiono le componenti delle normali, queste devono essere calcolate dal valore dell'intensità dei voxels del volume. Il metodo qui utilizzato è quello del gradiente d'intensità, già utilizzato da molti autori. Per ogni voxel del volume ricampionato le componenti del gradiente vengono approssimate con differenze finite centrali valutate fra i voxels confinanti con quello considerato per ciascuna delle tre direzioni principali; le stesse sono successivamente normalizzate dividendole per il modulo.

Poichè già all'origine le dimensioni del voxel non sono generalmente le stesse sulle tre dimensioni e, anche se così fosse, quasi certamente non lo sono dopo il ricampionamento del volume, le componenti del gradiente sono state calcolate dividendo la differenza fra le intensità per il doppio della dimensione fisica del voxel ricampionato.

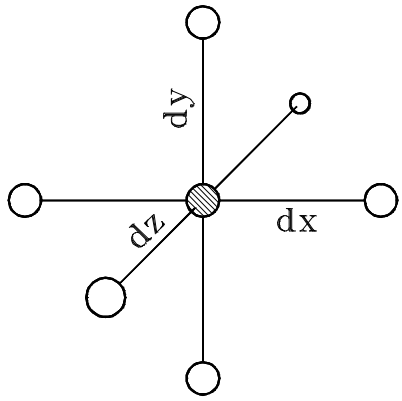

Figura 4: Determinazione del gradiente díntensità.

$$\nabla_{\mathrm{x}} f(\boldsymbol{x}_{\mathrm{i}}) = \frac{1}{2dx}\left[f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k)\right];$$

$$\nabla_{\mathrm{y}} f(\boldsymbol{x}_{\mathrm{i}}) = \frac{1}{2dy}\left[f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k)\right];$$

$$\nabla_{\mathrm{z}} f(\boldsymbol{x}_{\mathrm{i}}) = \frac{1}{2dz}\left[f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1})\right].$$

$$\mathrm{N}(\boldsymbol{x}_{\mathrm{i}}) = \frac{\nabla f(\boldsymbol{x}_{\mathrm{i}})}{\left|\nabla f(\boldsymbol{x}_{\mathrm{i}})\right|}$$

Una differenza fra il metodo di calcolo qui utilizzato e quello esposto da molti autori si riscontra in corrispondenza delle facce del volume. La considerazione che normalmente viene fatta è che, trattandosi di voxels di confine, le differenze centrali devono essere sostituite con differenze all'indietro o in avanti, a seconda della faccia considerata, in modo che i voxels coinvolti siano effettivamente appartenenti al volume. Se però si considera un volume completamente 'pieno' e omogeneo (il volume è un parallelepipedo i cui voxels hanno tutti lo stesso valore di intensità) e lo si immagina sospeso nel vuoto, come in realtà è mostrato dal visualizzatore, ci si rende conto che gli unici voxels con gradiente non nullo sono proprio quelli di confine per i quali il metodo delle differenze 'laterali' continua a dare gradiente nullo.

Il risultato corretto si ottiene allora continuando ad applicare anche sul confine le differenze centrali ed estendendo virtualmente il volume all'esterno con uno strato di voxels di intensità nulla.

Un'altra considerazione deve essere fatta circa il verso del gradiente. I sistemi grafici utilizzano solitamente il segno del prodotto scalare per determinare se la faccia della superficie che si sta osservando è quella anteriore o quella posteriore. Nel secondo caso la faccia non è visibile e non viene illuminata. Ciò funziona correttamente se la superficie considerata è opaca ma non, come invece in questo caso, se si tratta della superficie di separazione fra due sostanze di densità differente ma entrambe traslucide. Il segno del prodotto scalare deve essere allora interpretato unicamente come indicazione del fatto che detta superficie di separazione è vista attraverso il mezzo più denso; la superficie è quindi visibile e deve essere illuminata ugualmente. Conseguentemente l'algoritmo di calcolo delle

normali qui implementato considera unicamente la direzione del gradiente, non il verso.

Se si considera che i volumi da visualizzare sono uno per ogni direzione principale e che per ciascuno di essi l'illuminazione è sempre allineata con tale direzione, si comprende che una sola delle componenti della normale ha effetto sull'illuminazione del volume che le corrisponde. Dal calcolo delle normali sono pertanto generati 3 volumi distinti per le tre componenti.

Una osservazione deve essere fatta riguardo al modo in cui l'algoritmo tratta la forma indeterminata corrispondente all'annulamento del gradiente in zone a intensità costante. A differenza di altri algoritmi che associano a tali voxels componenti del gradiente nulle, si è qui deciso di assegnare alle stesse il valore massimo. Il motivo di tale decisione è conseguente al fatto che l'effetto visivo causato dall'adozione di componenti nulle sarebbe quello di oscurare completamente le parti 'piene' del volume quando invece le stesse devono contribuire positivamente alla visualizzazione dello stesso. La scelta è coerente con il modo in cui il volume è visto dai metodi di visualizzazione basati sulla classificazione: per questi esso è infatti una massa gelatinosa traslucida di densità variabile (maggiore là dove più alto è il valore dell'intensità) che riflette parte della luce incidente dove il gradiente è non nullo e diffonde la stessa nelle zone a densità costante. La corretta visualizzazione delle zone interne è garantita dall'applicazione dell'alpha blending che ne determina il livello di illuminazione in relazione all'assorbimento degli strati superficiali.

5.4 Classificazione

I metodi di classificazione presi in considerazione sono quelli, già ampiamente descritti in letteratura, di Levoy e di Drebin. Entrambi i metodi sono applicati utilizzando la possibilità offerta da OpenGL di impiegare le informazioni contenute nelle textures come valori d'ingresso in tabelle di look-up piuttosto che come valori di colore veri e propri. Impostando quattro tabelle distinte per le componenti R, G, B, A secondo i criteri descritti da Levoy o da Drebin è possibile modificare la classificazione in tempo reale semplicemente modificando le tabelle piuttosto che intervenendo direttamente sulle textures.

L'implementazione attuale del visualizzatore utilizza il modello di Drebin, generando, durante la fase di inizializzazione, una versione di default della tavola nella quale vengono fissati i colori e il range di intensità da associare a ciascun tessuto. Tanto i colori quanto il range di intensità possono successivamente essere manipolati interattivamente dall'osservatore mediante un semplice pannello di controllo. Sulla finestra superiore di questo compare un diagramma costituito da cinque bande associate ad altrettanti tipi di tessuto; agendo sull'altezza di una banda mediante il puntatore del mouse è possibile regolare l'opacità del tessuto corrispondente. Di ciascuna banda è possibile regolare anche l'ampiezza e la posizione in senso orizzontale, controllando così la gamma di intensità comprese in un dato tessuto, nonchè la pendenza delle transizioni con le bande adiacenti. Sulla finestra inferiore del pannello compaiono tre sliders che consentono di regolare a piacere le componenti HSV di colore di ciascun tessuto; il tessuto sul quale agiscono gli sliders è selezionato premendo il corrispondente radio-button situato al centro del pannello.

Le variazioni apportate tramite il pannello di classificazione sono visualizzate in tempo reale sul volume consentendo all'osservatore di raggiungere in breve tempo la classificazione che meglio evidenzia la regione di interesse del volume.

5.5 La struttura volume

Ultimata la fase di preprocessazione, tutte le caratteristiche del volume sono inserite in

una struttura denominata appunto *volume*.

Il volume contiene i puntatori alle sequenze di slices relative alle tre direzioni di vista, i puntatori ai tre volumi di componenti delle normali, le dimensioni fisiche del volume, la risoluzione del volume nelle tre direzioni (una sola informazione per la risoluzione delle slices che sono quadrate) un puntatore ad una struttura *cmap* contenente la dimensione delle color-maps e i quattro puntatori alle tavole R, G, B, A, un booleano che indica se le color maps sono in uso.

Nel volume sono inserite successivamente, durante l'inizializzazione, altre informazioni necessarie al rendering: una tavola 3×2 contenente l'indice della prima e dell'ultima texture per le tre direzioni di vista; un campo 'posizione osservatore' che definisce quale sia la direzione di vista corrente e quale faccia del volume sia posizionata in alto rispetto all'osservatore, una tavola che definisce le dimensioni del volume in funzione della posizione dell'osservatore. L'uso di queste informazioni aggiuntive è spiegato più avanti.

6. Il rendering

6.1 Display lists

L'uso delle display lists rappresenta uno dei modi per massimizzare l'efficienza di una sistema grafico OpenGL. Racchiudendo in una list una sequenza di chiamate alla libreria si ottiene l'effetto (in maniera variabile a seconda del tipo di chiamate contenute) di 'tradurre' le chiamate in una forma ottimizzata per l'hardware grafico. Particolarmente vantaggioso è il loro uso per l'accumulazione di matrici di trasformazione e per l'assegnazione delle textures del volume. Nel primo caso vengono generate, durante la compilazione della lista, sia la matrice di trasformazione composta che la sua inversa cosicchè una chiamata della lista durante l'esecuzione richiede un tempo dell'ordine di una semplice LoadMatrix. Nel secondo caso l'uso di display lists è invece richiesto dal sistema per caricare le textures anticipatamente su una memoria appositamente predisposta. La possibilità di organizzare le lists sequenzialmente consente inoltre di snellirne la gestione e di alleggerire il codice. Ad esempio, le tre serie di textures corrispondenti alle tre viste del volume sono raggruppate in tre sequenze di liste con indici crescenti procedendo dalla texture anteriore verso la posteriore; una semplice scansione sequenziale delle stesse (back-to-front o front-to-back a seconda che il volume sia osservato anteriormente o posteriormente) consente di posizionare le slices nell'ordine corretto.

Altri due gruppi di liste (posizionamento textures e posizionamento osservatore) sono utilizzate per posizionare sempre correttamente le textures sui piani di supporto e i piani stessi nello spazio in funzione della posizione di vista. In questo caso gli indici delle liste sono ordinati sequenzialmente ma la determinazione della lista da utilizzare viene fatto, come illustrato in dettaglio più avanti, mediante una semplice struttura di supporto linkata e mediante tabelle.

Tutte le textures sono precompilate durante la fase di inizializzazione (immediatamente dopo le operazioni di resampling del volume, calcolo delle normali e preparazione delle color-maps) riducendo così al solo indirizzamento l'overhead connesso al loro uso durante il rendering.

6.2 Posizione, dimensione del volume e strutture di supporto

Disegnare un piano nello spazio e specificare la texture da incollarci sopra viene fatto in OpenGL specificando le coordinate dei vertici di entrambi. Una delle caratteristiche principali di OpenGL è che le coordinate ricevute sono sempre assunte come coordinate in object space (o texture space) e riportate in coordinate di vista mediante una opportuna matrice di trasformazione (la matrice di ModelView per i vertici dei piani, quella di texture per le coordinate di texture). Anche se le coordinate fossero riferite direttamente al sistema dell'osservatore verrebbe eseguito comunque un prodotto per la matrice di identità, come dire che i due sistemi di riferimento sarebbero sempre differenti anche se coincidenti. Per questo motivo non ha praticamente alcun senso determinare via software l'esatta posizione dei vertici rispetto all'osservatore quando questa può essere assegnata rispetto al sistema di riferimento più conveniente (quello che consente la minor quantità di calcoli) e trasformata dall'hardware del sistema.

La scelta è stata quindi quella di vedere il volume come un cubo di dimensioni unitarie centrato sull'origine del proprio sistema di riferimento (vertici opposti [-0.5, -0,5, -0.5] [0.5, 0.5, 0.5]) e di impostare la matrice di modelview in modo da eseguire, per ogni direzione di vista, tre scalature lungo i tre assi che ne riportano le dimensioni a quelle reali e una rotazione intorno all'asse Z che seleziona l'alto del volume. Il tempo di impostazione della matrice e ridotto al minimo dall'uso delle display lists.

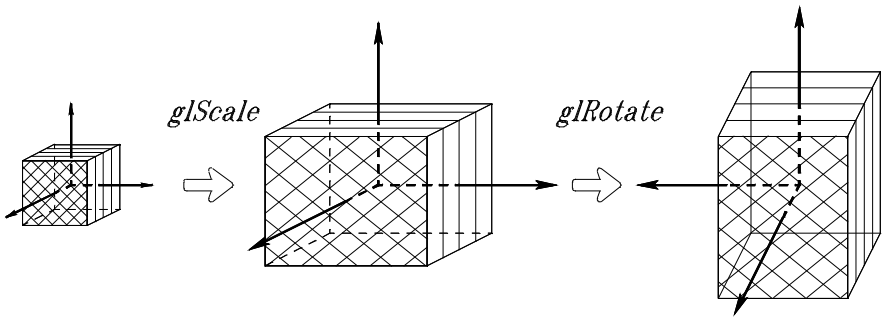

Figura 5: Riproporzionamento e posizionamento del volume.

Per ogni direzione di vista l'algoritmo di rendering disegna una sequenza di piani di dimensioni unitarie, a partire dalla posizione z = -0.5 verso quella z = 0.5, incollando su essi le textures nell'ordine inverso a quello col quale sono mantenute nella lista; il sistema trasforma i piani secondo la matrice di modelview provocando di conseguenza lo stiramento delle textures.

Il cambiamento della direzione di vista non modifica assolutamente il tracciamento dei piani ma semplicemente la serie di textures da utilizzare e la distribuzione dei fattori di scalatura sui tre assi. Per evitare di dover determinare di volta in volta queste informazioni si è fatto uso di una struttura 'linkata' che tiene traccia della posizione relativa volume-osservatore e di tabelle degli indici delle liste che contengono le impostazioni relative; dalla struttura di posizione sono prelevati i valori d'ingresso nelle tabelle e da queste sono estratti gli indici delle liste che devono essere chiamate per attivare le impostazioni.

La struttura è creata in fase di inizializzazione del volume e nel volume stesso è mantenuto un puntatore alla posizione corrente sulla struttura; tale puntatore è aggiornato ogni volta che l'utente del visualizzatore modifica la posizione dell'oggetto.

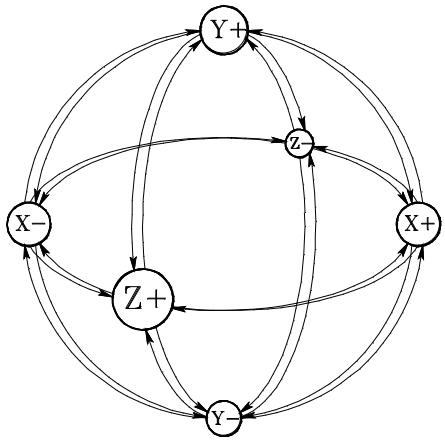

Figura 6: Struttura di tracciamento della posizione di vista.

Ogni nodo della struttura contiene il nome simbolico dell'asse di vista corrente, la posizione rispetto all'origine (semiasse positivo o negativo) e quattro puntatori alle posizioni di vista confinanti. Nel volume, insieme al puntatore alla posizione corrente sulla struttura è mantenuto anche il nome simbolico del confinante situato a nord. L'insieme di queste informazioni consente di conoscere esattamente quale posizione sarà assunta dall'osservatore in seguito ad uno spostamento sulle quattro direzioni cardinali e quale dei nuovi confinanti si troverà a nord. Poichè le textures sono generate per ciascuna direzione in modo che il loro sistema di riferimento sia sempre orientato nel verso positivo di due assi del volume e con la normale diretta come il terzo asse (ciò corrisponde a dire che se l'osservatore è posto nel semispazio positivo definito dal piano YZ gli assi X ed Y delle textures coincidono rispettivamente con gli assi Y e Z del volume e la normale con l'asse X dello stesso) la posizione del nord dopo un qualunque spostamento sulla struttura è funzione sia della sua posizione precedente che dell'orientazione del nuovo set di textures; il nuovo nord viene pertanto desunto da apposite tabelle utilizzando come ingresso il valore precedente allo spostamento.

6.3 Posizione delle textures

Le coordinate di texture sono sempre comprese fra 0 e 1 (0,0 l'angolo inferiore sinistro, 1,1 quello superiore destro) mentre quelle del piano che le supporta possono assumere un valore qualunque all'interno del campo di definizione dei numeri in floating point; è compito del programmatore eseguirne la corretta associazione. In questo caso bisognerebbe incollare lo spigolo 0, 0 della texture allo spigolo -0.5, -0.5 del piano e lo spigolo 1, 1 della prima sullo spigolo 0.5, 0.5 del secondo.

Questo non rappresenta certamente un problema per un caso così banale ma può invece appesantire l'algoritmo quando si vogliano eseguire esplicitamente delle associazioni fra punti intermedi della texture con punti intermedi del piano. Un caso del genere può verificarsi manipolando il volume: ad esempio incidendolo e discostando i lembi per osservarne l'interno. L'incisione dovrebbe essere materialmente praticata, in generale approssimandola con una spezzata più o meno fitta, tanto sui piani del volume che sulla texture, il che comporterebbe l'esecuzione dei cacoli necessari a riportare le coordinate dei punti della spezzata sul piano a quelle dei punti della spezzata sulla texture. Il peso di questi calcoli può essere considerevole in realzione alle dimensioni, all'irregolarità del taglio e alla sua profondità (numero dei piani interessati dal taglio).

La possibilità di trasformare le coordinate di texture attraverso la matrice corrispondente consente di risolvere agevolmente il problema: si specificano per la texture le stesse coordinate calcolate per il piano e si imposta la matrice di trasformazione in modo che questa esegua il riporto. La cosa non comporta alcun appesantimento delle operazioni eseguite dal sistema che in ogni caso moltiplicherebbe le coordinate almeno per la matrice d'identità.

Nel semplice caso di partenza, per riportare le coordinate -0.5, -0.5 e 0.5, 0.5 a 0, 0 e 1, 1 si dovrebbe caricare sul matrix stack una matrice di traslazione di 0.5 unità nel verso positivo di entrambi gli assi.

Ancora più utile è l'utilizzazione di una trasformazione per implementare la visione posteriore del volume. In questo caso infatti non è solo invertito l'ordine in cui le texture sono visualizzate ma sopratutto deve esserne visualizzata la parte posteriore. La soluzione adottata è quella di ruotare la texture di 180° intorno al proprio asse baricentrale verticale.

6.4 Fusione delle textures col piano

Come già detto nella sezione introduttiva, OpenGL mette a disposizione diverse modalità di incollaggio delle textures con la superficie. La modalità qui utilizzata è la GL_MODULATE in base alla quale il colore e l'opacità risultanti per il pixel sono ricavate applicando le relazioni:

$$C_p = C_t C_f ,$$

$$A_p = A_t A_f ,$$

in cui i pedici p, t, ed f si riferiscono al pixel, al texel e al fragment (rispettivamente lo stesso elemento d'immagine per il display, per la texture e per il poligono), la lettera C indica una qualunque delle tre componenti di colore, la lettera A indica l'opacità.

Da queste relazioni risulta che fondendo le textures con piani aventi tutte le componenti unitarie si ottiene la riproduzione della texture senza alterazioni. E' quindi come se i piani fossero delle lastre di vetro assolutamente invisibili o come se si ricomponesse il volume riposizionando direttamente le slices originarie.

6.5 Distance only shading

Si è già fatto notare che il maggior difetto del texture mapping per il rendering di volume è l'impossibilità di ottenere una illuminazione corretta semplicemente abilitando il lighting di OpenGL. Questo, lo si ricorda, è dovuto al fatto che ciò che il sistema sta disegnando sono i piani che supportano le texture, non il contenuto delle stesse.

Non potendo valutare le reali componenti di riflessione diffusa e speculare, il visualizzatore deve limitarsi a tenere conto unicamente dell'attenuazione che l'intensità luminosa subisce nel senso della profondità per conferire tridimensionalità, seppure lieve, all'immagine.

Il risultato potrebbe essere raggiunto ancora abilitando il lighting, adottando su tutti i vertici dei poligoni normali dirette verso la sorgente di luce, utilizzando un coefficiente di riflessione diffusa pari a 1 e un coefficiente di riflssione speculare nullo; si otterrebbe così di annullare, nel modello d'illuminazione applicato dal sistema, il peso dei termini relativi alla riflessione. Il sistema dovrebbe comunque eseguire almeno un prodotto scalare per ogni vertice, un peso troppo elevato per risultati così modesti.

Un metodo alternativo potrebbe essere quello di abilitare l'applicazione delle equazioni di fog:

$$C_o = C_i f + C_f (1 - f),$$

$$f = \frac{end - z}{end - start},$$

con C_o componente di colore uscente, C_i componente di colore di partenza, f coefficiente di fog dato dalla seconda equazione. In quest'ultima, che rappresenta la possibilità di minor complessità computazionale fra quelle disponibili in OpenGL, *end* e *start* rappresentano gli estremi entro i quali può variare *z* (distanza nel senso di profondità); essi dovrebbero essere impostati sui punti estremi del volume per ottenere il massimo effetto di tridimensionalità.

L'utilizzazione del fog comporterebbe sicuramente una efficienza superiore rispetto al metodo precedente e consentirebbe ancora una corretta illuminazione (sempre rimanendo nei limiti della semplicità del modello) per qualunque posizione della sorgente.

Ancora diverso è il metodo adottato dal visualizzatore per implementare il distance-only shading; esso si basa sulla citata ipotesi che la sorgente d'illuminazione sia in posizione fissa rispetto al volume e che la direzione d'illuminazione sia ortogonale alla sua faccia anteriore. Sebbene tale limitazione sia strettamente necessaria solo per il passaggio alla successiva modalità d'illuminazione (il normal shading), essa si dimostra utile anche in questo caso consentendo una ulteriore semplificazione delle operazioni di calcolo. Le limitazioni imposte garantiscono infatti che ogni piano sia illuminato uniformemente su tutta la superficie e che una differenza nel livello d'illuminazione possa riscontrarsi solo fra un piano e l'altro. Ipotizzando una attenuazione lineare e considerando la spaziatura uniforme fra i piani del volume, risulta che il decremento d'illuminazione fra un piano e il successivo (o meglio l'incremento dato che l'algoritmo lavora in back-to-front) è costante e il suo aggiornamento richiede semplicemente una somma.

Poichè, qualunque sia il modello d'illuminazione adottato, una volta determinata la luce incidente in un punto dell'oggetto il sistema calcolerebbe il colore finale C_f eseguendo il prodotto del colore C_s della prima per il colore C_o del secondo:

$$C_f = C_s C_o,$$

osservando che questa è anche l'operazione che viene eseguita fra il colore della texture e quello del piano durante la fusione, se ne deduce che è possibile implementare il modello di illuminazione distance-only semplicemente regolando l'intensità di colore dei piani di supporto delle textures. Il visualizzatore assume quindi che il colore dell'ultimo piano del volume sia quello definito per la componente di luce ambiente e che quello del primo sia definito dal colore della sorgente; dividendo poi la differenza dei colori per il numero di intervalli (numero di piani - 1) determina l'incremento di colore fra due piani adiacenti.

6.6 Normal shading

Dal momento che non esiste modo per il sistema di applicare questo modello d'illuminazione con le textures, si è deciso di eseguirlo via software direttamente sul volume e di fornire al sistema delle textures già illuminate. Naturalmente ciò è corretto solo in quanto si è fissata la posizione relativa sorgente-volume e se si considera solamente la componente di riflessione diffusa e non quella speculare, la quale dipende anche dalla posizione dell'osservatore.

Considerando nulla la componente di luce ambiente, dal modello d'illuinazione e dall'applicazione della luce incidente al colore del punto dell'oggetto illuminato si ottiene:

$$C_f = C_o \cdot k_{att} \frac{end - z}{end - start} \cdot k_d \mathbf{N} \times \mathbf{L}$$

dove C_f è il colore risultante, C_o è il colore dell'oggetto (in questo caso il texel della texture), k_{att} è il fattore di attenuazione (la quale si suppone dipenda ancora linearmente dalla distanza dalla sorgente), k_d è il coefficiente di riflessione diffusa del corpo e l'ultimo termine il noto prodotto scalare. Considerando ancora la relazione di fusione del colore del texel con quello del piano, si arriva alla conclusione che è ancora possibile valutare il termine di attenuazione nella colorazione del piano e, ponendo a 1 il coefficiente di riflessione diffusa, determinare il colore del texel illuminato C_{tf} semplicemente come prodotto del colore proprio C_{to} per il risultato del prodotto scalare:

$$C_{tf} = C_{to} \mathbf{N} \times \mathbf{L}.$$

Ancora si deve osservare che l'aver fissato una direzione d'illuminazione ortogonale ai piani del volume comporta che il prodotto scalare dipenda dalla sola componente della normale in tale direzione, componente che è stata memorizzata, durante il preprocessing, nel volume associato alla direzione stessa. Lo shading si riduce dunque a moltiplicare, per ciascuna direzione di vista, il colore dei texels delle textures per il corrispondente elemento del volume delle normali. Naturalmente bisogna tenere conto del fatto che nelle textures sono contenuti inizialmente solo i valori delle intensità che devono essere convertiti in terne R,G,B utilizzando le color-maps. Dopo lo shading si disporrà quindi di textures con elementi di quattro componenti (R, G, B, A, di cui l'ultimo è stato caricato dalla relativa tavola durante lo shading stesso) che saranno passate al sistema grafico in sostituzione di quelle precedenti.

Sebbene il metodo sia corretto, è tuttavia presente il difetto che, avendo considerata nulla la componente di luce ambiente, le parti del volume con normale ortogonale alla direzione d'illuminazione sono completamente buie. Purtroppo però, per il modo in cui OpenGL fonde la texture col piano e per il fatto che le textures possono essere visualizzate in entrambi i sensi a seconda della posizione del volume rispetto all'osservatore (e alla sorgente di luce), non è possibile far coesistere una componente di luce diretta e attenuata con una componente di luce ambiente assolutamente costante.

La soluzione che ha portato a risultati visivi più che soddisfacenti è stata quella di considerare la sorgente di luce in parte direzionale ed in parte diffusa. La situazione è analoga a quella ampiamente sfruttata dai fotografi che a sorgenti direzionali abbinano ampi pannelli diffusori illuminati al fine di addolcire le ombre generate dalle prime. Se il pannello è sufficientemente ampio e posizionato come la sorgente esso produce una illuminazione indipendente dalla orientazione delle normali ma ancora soggetta ad attenuazione nel senso della profondità.

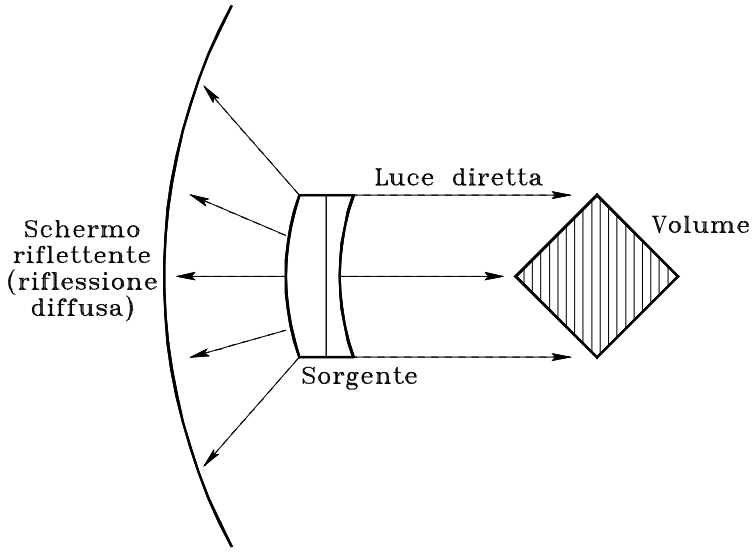

Figura 7: Modello di illuminazione.

L'adozione di questo modello non comporta infine nessuna variazione anche quando è applicato il distance-only shading.

6.7 Correzione dell'opacità

Si è già detto che l'utilizzazione di textures piane con proiezioni di tipo prospettico crea una apparente variazione dell'opacità delle textures dal centro (o meglio dal piede della perpendicolare condotta per l'osservatore) verso i bordi. Su un sistema che supporta le texture tridimensionali, la soluzione più semplice sarebbe quella di utilizzare settori sferici anzichè piani; penserebbe il sistema a ritagliare correttamente le textures dal volume per qualunque direzione di vista. In questo caso però non si può fruire di questa agevolazione.

La soluzione qui adottata è invece quella di suddividere i piani del volume in un numero prefissato di poligoni ed eseguire sui loro vertici la correzione dell'opacità in base all'angolo d'incidenza dei raggi di vista.

A regola, la correzione di opacità dovrebbe essere eseguita sui texels delle textures.

Per determinare la nuova opacità si consideri una slice di spessore t_1 appartenente ad un volume omogeneo di opacità a; per una slice di spessore t_2 la sua opacità potrebbe essere dedotta dalla precedente utilizzando la relazione:

$$a_{t2} = 1-(1-a_{t1})^{t2/t1}.$$

Ma t_2 può anche essere visto come la lunghezza del percorso necessario ad attraversare la prima slice diagonalmente. Se α è l'angolo che il raggio incidente forma con la normale alla slice, l'opacità risultante è data da:

$$a_{t2} = 1-(1-a_{t1})^{1/cos\alpha}.$$

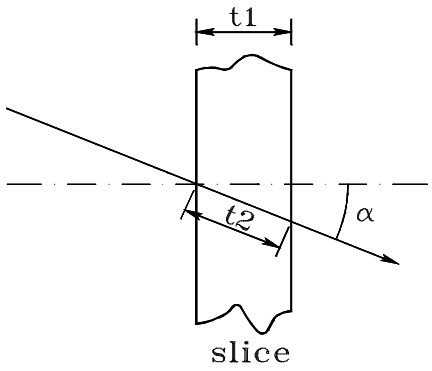

Figura 8: Variazione di opacità con l'angolo di incidenza.

Purtroppo, a causa della continua variazione della posizione di vista, la precedente relazione non può essere applicata una volta per tutte alle textures, nè si può pensare di ricalcolare le stesse ad ogni spostamento dell'osservatore.

Ricordando come il sistema determina l'opacità finale durante l'operazione di fusione texture-piano,

$$A_p = A_t \cdot A_f',$$

si può però pensare di risolvere il problema intervenendo sulla opacità del secondo, moltiplicandola per un fattore di correzione:

$$A_f' = k_\alpha(A_t, \alpha) \cdot A_f,$$

dove ancora i pedici *p*, *t* ed *f* si riferiscono rispettivamente al pixel, al texel e al fragment, mentre k_a è il supposto fattore di correzione dell'opacità.

Ancora si presenta il problema che il calcolo di A'_f deve essere eseguito via software prima di tracciare il piano, cioè ben prima che il sistema esegua la fusione, quando non è ancora noto il valore A_t da cui k_a dipende.

La soluzione adottata è stata quella di approssimare $k_a(A_t, \alpha)$ con una funzione del solo angolo d'incidenza del raggio di vista. Per fare ciò è stata eseguita la correzione di opacità per una ampia gamma di angoli d'incidenza su un gran numero di valori di opacità del texel, è stato calcolato il rapporto fra le due opacità ed è stata determinata una funzione che approssimasse al meglio tale rapporto:

$$k_\alpha(\alpha) = \left(\frac{1}{cos\varphi}\right)^{cos\varphi}.$$

L'approssimazione raggiunta è molto buona per valori di opacità del texel compresi fra 0 e 0,5 e si riduce progressivamente all'aumentare dell'opacità.

Lo scarto è inferiore all' 1% fino ad α=0,5, inferiore al 5% per α=0,6, diviene del 13% per α=0,8 e del 27% per α=1,0; bisogna tuttavia osservare che questi scarti si riferiscono ad angoli d'incidenza di 45°, mediamente superiori a quelli che possono aversi durante il normale funzionamento del visualizzatore, e che per angoli inferiori si riducono

sensibilmente (a 30° lo scarto si riduce al 13% per α=1,0).

Un modo per incrementare la precisione del metodo può essere quello di ridurre l'opacità iniziale delle textures riducendo lo spessore delle slices del volume. Ad esempio, bisezionando una slice di opacità 0,9 se ne otterrebbero due di opacità 0,68 valore per il quale lo scarto è inferiore al 10% per angoli di 45° e al 6% per angoli di 30°. Il rovescio della medaglia è che ciò comporta la gestione del doppio delle textures col conseguente raddoppio dell'occupazione di memoria e dei tempi d'esecuzione.

Fortunatamente non è necessario adottare questa soluzione grazie al fatto che il sistema grafico esegue il clamping a 1,0 dei valori di opacità superiori all'unità; ciò comporta una compressione degli scarti proprio là dove sono maggiori (ad esempio, incrementando α=0,9 del 20% invece che del 5% si otterrebbe α=1,08 invece che α=0,945; grazie al clamping lo scarto si ridurrebbe dal 14% al 5,8%).

Per evitare di dover eseguire il calcolo di $k_a(\alpha)$ durante il rendering (il peso del calcolo di una funzione trigonometrica e di un elevamento a potenza reale è inaccettabile) la funzione è calcolata in fase di inizializzazione e tabellata mappando i valori di $cos\varphi$ sull'intervallo 0-255; durante l'esecuzione se ne può quindi ricavare il valore al solo prezzo di una moltiplicazione e di un accesso alla tavola.

Rimane ancora il problema di determinare il valore di $cos\alpha$ per tutti i vertici dello strip di quadrilateri in cui il piano è suddiviso. La sua determinazione richiede infatti il calcolo di un prodotto scalare (normale al piano $\times$ direzione del raggio di vista) per ciascuno di tali punti, ben 1600 prodotti scalari per fotogramma (per ciascuno dei quali sono necessari, trascurando le operazioni di peso minore, perlomeno tre quadrati e una radice quadrata) con soli 64 piani di texture e 4 suddivisioni.

La soluzione più semplice è ancora una volta quella di ricavare i suddetti valori per interpolazione lineare.

Per una data posizione dell'osservatore si calcolano i prodotti scalari sugli otto vertici del volume e se ne determina, sui quattro spigoli nel senso della profondità, l'incremento fra piani adiacenti; l'incremento è costante per l'equidistanza dei piani. Procedendo nell'algoritmo di visualizzazione dall'ultimo piano verso il primo, si aggiornano i coseni, sui quattro vertici di ciascuno di essi, per semplici incrementi. Per ogni piano si ripete l'operazione determinando gli incrementi lungo gli spigoli verticali e, avanzando verso l'alto per file di vertici della scomposizione, si determinano ancora gli incrementi sulla riga di vertici e i valori sui vertici per somma. Gli incrementi sono costanti sia in senso verticale che in senso orizzontale per l'uniformità della suddivisione; il loro calcolo richiede una differenza ed una moltiplicazione.

In questo modo devono essere svolte 24 moltiplicazioni, 8 divisioni e 8 radici quadrate iniziali per i prodotti scalari sui vertici del volume e, considerando ancora 64 piani di texture e 4 suddivisioni, 452 moltiplicazioni e 2044 somme per il loro aggiornamento.

Le operazioni più pesanti, quelle iniziali, non dipendono ora dalle suddivisioni fissate; inoltre l'adozione di alcuni accorgimenti ha consentito di limitarne il costo a quello delle sole 8 divisioni e di 32 moltiplicazioni.

Osservando infatti che la normale ai piani di texture è diretta come l'asse Z del volume e che la posizione di vista è data rispetto allo stesso sistema di riferimento, si ha che il prodotto scalare si riduce al quoziente della componente su Z del vettore rappresentante la posizione dell'osservatore (naturalmente riferita al vertice sul quale il prodotto è valutato mediante semplice traslazione del sistema) per il modulo dello stesso. Elevando al quadrato numeratore e denominatore (la qual cosa non comporta operazioni aggiuntive dato che il quadrato del numeratore compare già a denominatore) ed eseguendone il rapporto si ottiene il vantaggio di dover estrarre la radice quadrata di un numero compreso fra 0 e 1. E' quindi facile mappare questo valore sull'intervallo 0-255 e ricavare il risultato cercato da una piccola tavola di look-up appositamente predisposta in fase di inizializzazione. Il peso delle 8 radici quadrate è così ridotto a quello di 8 moltiplicazioni e altrettanti accessi alla tavola.

In realtà il calcolo che il visualizzatore esegue è leggermente più complesso del

precedente. Si deve tenere in cosiderazione infatti che non necessariamente la variazione dell'angolo d'incidenza dei raggi di vista è una funzione monotona su tutta l'estensione del volume; ciò accade solamente se la perpendicolare condotta dall'osservatore ai piani dello stesso cade al di fuori delle fasce di cui ciascun piano è intersezione. Tuttavia questa condizione non è assicurata, anzi nella maggior parte dei casi accadrà il contrario.

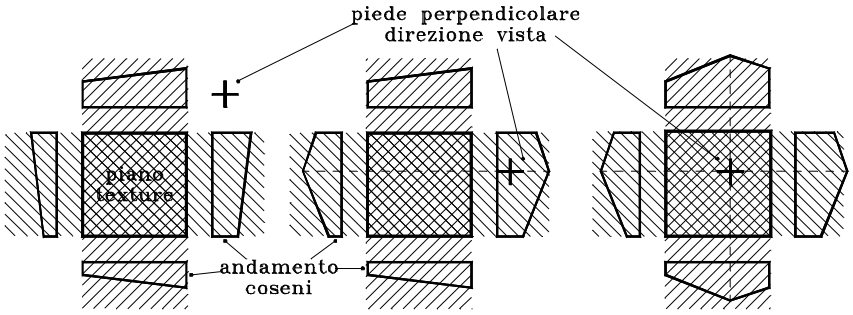

Figura 9: Andamento del cosα in funzione del punto di osservazione.

L'algoritmo effettivamente implementato è molto simile al precedente ma il volume è suddiviso in quattro parti prorpio in corrispondenza del piede della perpendicolare per il punto di vista. In questo caso 16 diventano i vertici di partenza e doppi saranno gli incrementi da valutare per ogni lato del piano nel caso più sfavorevole.

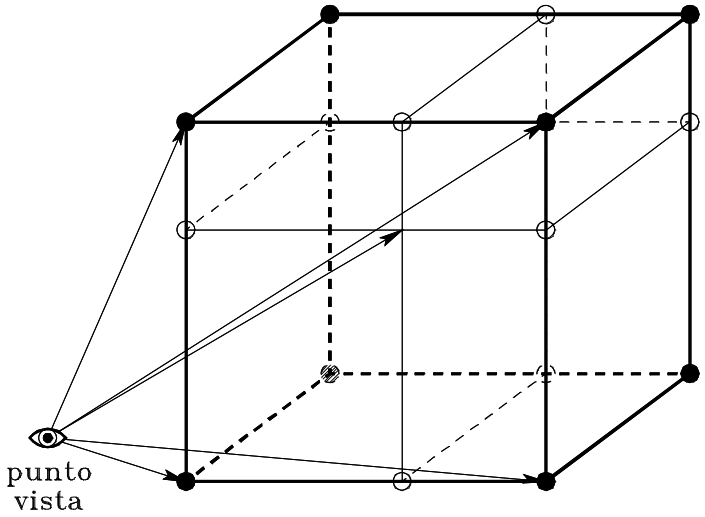

Figura 10: Valutazione incrementale del cosα a partire dal valore sui vertici.

Finalmente, nota l'opacità sui vertici dello strip di quadrilateri (in realtà tringles strips per la implementazione corrente del visualizzatore), è lasciato al sistema grafico l'onere di eseguirne l'interpolazione sui punti interni a ciscun quadrilatero; questa caratteristica è nota come *smooth shading* ed ha tempi d'esecuzione estremamente contenuti grazie alla implementazione hardware.

7. Stereoscopia e tracking

7.1 Tracking [9]

Ciò che il sistema di tracking deve rilevare è la posizione degli occhi dell'osservatore rispetto al piano di vista.

Il sistema di tracking utlizzato è costituito essenzialmente da tre rilevatori posizionati sulle stanghette e sul ponticello degli occhiali che l'osservatore indossa e da tre emettitori solidali col monitor; mediante triangolazione del segnale captato dai rilevatori il sistema risale alla posizione degli emettitori che individuano la posizione del sistema di riferimento degli occhiali.

Richiedere al sistema le informazioni di posizione rilevate è analogo a richiedere le coordinate di un mouse o altro dispositivo locator tradizionale; è invece compito del programmatore riportare queste coordinate al proprio sistema di riferimento. Particolarmente importante è, in questa fase, la taratura del sistema. Le coordinate restituite dal sistema di tracciamento indicano infatti solo la posizione degli occhiali rispetto al rilevatore ma l'effettiva posizione dell'osservatore rispetto al monitor (ciò che veramente interessa) dipende sia da come gli occhiali sono indossati dal primo che da come il rilevatore è posizionato sul secondo. Il programma che utilizza le informazioni di tracking deve perciò predisporre delle operazioni preliminari di taratura atte a collimare i due sistemi di riferimento. La procedura di calibrazione prevista nel TVR è quella già descritta in precedenza.

Le unità di misura non pongono particolari problemi: utilizzando le dimensioni fisiche (ad esempio espresse in centimetri o millimetri) degli elementi coinvolti nella visualizzazione (dimensioni del monitor, distanza dell'osservatore dal monitor e distanza interoculare) è semplicemente necessario rappresentare anche gli oggetti dell'immagine con le proprie dimensioni reali espresse nella stessa unità di misura. Naturalmente bisogna ricordarsi di adottare l'opportuno fattore di scala se la dimensione reale degli oggetti è tale da non poter essere contenuta nel display.

7.2 Stereoscopia [10][11][12]

L'implementazione della visione stereoscopica non richiede particolari accorgimenti. Nota la posizione degli occhi dell'osservatore rispetto al display sono definiti i volumi di vista (*frustums*) come le piramidi aventi il display per base e un occhio per vertice; la base della piramide è anche finestra di vista in quanto su essa sarà proiettata l'immagine.

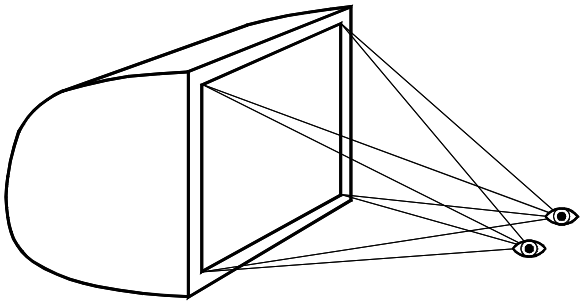

Figura 11: Volumi di vista distinti per i due occhi.

Sulla stessa finestra di vista devono dunque essere proiettate due immagini ognuna delle quali è però destinata solo ad un occhio. Per riuscire ad associare l'immagine all'occhio per il quale è disegnata, il sistema grafico mette a disposizione dell'applicazione due frame buffers distinti, un *left buffer* e un *right buffer*; l'applicazione genera l'immagine relativa al volume di vista sinistro e la invia al left buffer poi ripete l'operazione per il volume di vista destro e invia l'immagine al right buffer. Il sistema grafico invia quindi al display alternativamente le immagini contenute nei due buffers. Naturalmente deve esistere un modo per consentire all'osservatore di ricevere le due immagini separatamente sui due occhi, altrimenti egli vedrebbe una immagine confusa nella quale si sovrappongono due viste con angolazioni diverse dello stesso oggetto. La cosa è resa possibile da occhiali speciali con lenti a cristalli liquidi: questi sono dotati di un ricevitore che, captato un segnale inviato dal sistema, comanda l'oscuramento di una lente per volta. Il sistema dispone invece di un emettitore azionato in sincronia con la commutazione del frame buffer: durante la visualizzazione della immagine sinistra è oscurata la lente destra e viceversa. L'osservatore percepisce così la profondità dell'oggetto esattamente come se si trovasse di fronte all'oggetto reale.

La posizione effettivamente assunta dall'oggetto rispetto allo schermo dipende da come l'applicazione ha posizionato questo nel prorpio sistema di riferimento. Poichè tale sistema ha l'origine nel centro dello schermo, l'oggetto apparirà per metà fuori e per metà dentro il monitor se l'applicazione l'ha centrato sull'origine, viceversa sarà completamente fuori dal monitor se è stato traslato sull'asse Z di almeno metà della sua profondità.

Per poter esegure la manipolazione dell'oggetto è quindi opportuno che questo sia posizionato interamente al di fuori dello schermo, solo così l'osservatore può riuscire a 'toccarlo' in tutte le sue parti.

La condizione migliore per una semplice osservazione senza manipolazione è tuttavia la prima. Il primo motivo è che più l'oggetto è avanzato rispetto allo schermo più la sua immagine si sposta rapidamente verso i bordi dello stesso quando l'osservatore si sposta lateralmente; come l'immagine va al di là di uno dei bordi una parte dell'oggetto scompare.

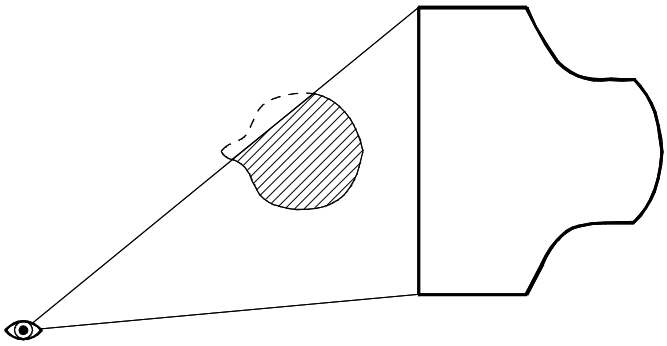

Figura 12: Uscita di parte dell'oggetto virtuale dal volume di vista.

Il secondo motivo è che l'osservatore si trova a mettere a fuoco una immagine visualizzata sullo schermo pur osservando, apparentemente, qualcosa che gli è più vicino; sebbene lo sforzo necessario a superare questa incongruenza sia modesto, una visione prolungata può portare ad un certo affaticamento.

8. Le prestazioni

Nonostante il visualizzatore sia stato realizzato per funzionare su una ampia gamma di piattaforme, il suo sviluppo è stato eseguito interamente su una stazione grafica *Infinite Reality* della Silicon Graphics dotata di due processori MIPS R10000 a 194 Mhz con relativi coprocessori per le operazioni in floating point, 1Gb di memoria RAM e due banchi di memoria di texture di 16Mb ciascuno. Il tracking dell'osservatore è eseguito mediante il sistema *Logitech Ultrasound* e gli occhiali *CrystalEyes* utilizzati, grazie alle lenti a cristalli liquidi, anche per la visione stereoscopica.

Grazie all'aggiunta di semplici routines di temporizzazione, è stato rilevato una frequenza fotogrammetrica di circa 10 immagini al secondo con una risoluzione del volume di 128^3 (128 textures di 128^2 pixels ciascuna), di 20 immagini al secondo con $128^2 \times 64$ di risoluzione (64 textures di 128^2) e di più di 30 immagini al secondo con 64^3 di risoluzione.

Queste prestazioni sono state ottenute attivando il normal shading mentre inferiori sono quelle riscontrate con l'attivazione del distance-only shading. La causa risiede nella utilizzazione, con l'uso del distance-only, delle tabelle di look-up per la classificazione del volume. Per ciascun fragment il sistema deve infatti convertire il valore estratto dai texels delle textures eseguendo 4 accessi in tabella prima di disporre della quaterna R, G, B, A da utilizzare per la fusione. E' sintomatico il fatto che le prestazioni raggiunte con i due metodi sono le stesse per basse risoluzioni delle textures (64^2) e divergono sensibilmente per risoluzioni più elevate; nel secondo caso infatti i tempi di manipolazione delle textures sono predominanti. A 128^3 il distance-only consente una frequenza di fotogramma di circa 5 immagini al secondo. Le prestazioni riportate si riferiscono a immagini che ricoprono una superficie massima della viewport stimata intorno ai 250000 pixels; per superfici superiori le prestazioni degradano sensibilmente a causa della saturazione dello stadio finale del pipeline di rendering. Queste dimensioni sono tuttavia tutt'altro che modeste se si pensa che 250000 sono i pixels ricoperti dall'immagine di una sfera il cui diametro è pressapoco la metà del lato di uno schermo quadrato che lavora con una risoluzione di 1280×1024; per uno schermo da 20' questo diametro è di circa 17 cm.

Fra gli inconvenienti riscontrati bisogna citare:

- Un certo affaticamento della vista dovuta alla discordanza fra la distanza di messa a fuoco e quella di convergenza delle direzioni di vista dei due occhi. Ciò è dovuto al fatto che l'osservatore, pur avendo l'impressione di osservare un oggetto collocato in posizione intermedia fra esso e lo schermo, deve mettere a fuoco delle immagini che in realtà sono situate sullo schermo stesso. Tale affaticamento può divenire sensibile se si osserva l'oggetto da distanza ravvicinata (si avverte anche un peggioramento dell'effetto stereoscopico) mentre è molto modesto (e l'effetto stereoscopico massimo) se la distanza dallo schermo non scende al di sotto di un metro circa.
- La tendenza del sistema ad evidenziare i contorni delle slices quando l'osservatore assume posizioni molto angolate rispetto alla perpendicolare condotta per il centro dello schermo. Tale effetto è però molto contenuto con 128^3 di risoluzione e se l'osservatore mantiene una distanza dallo schermo pari almeno a quella consigliata al punto precedente.

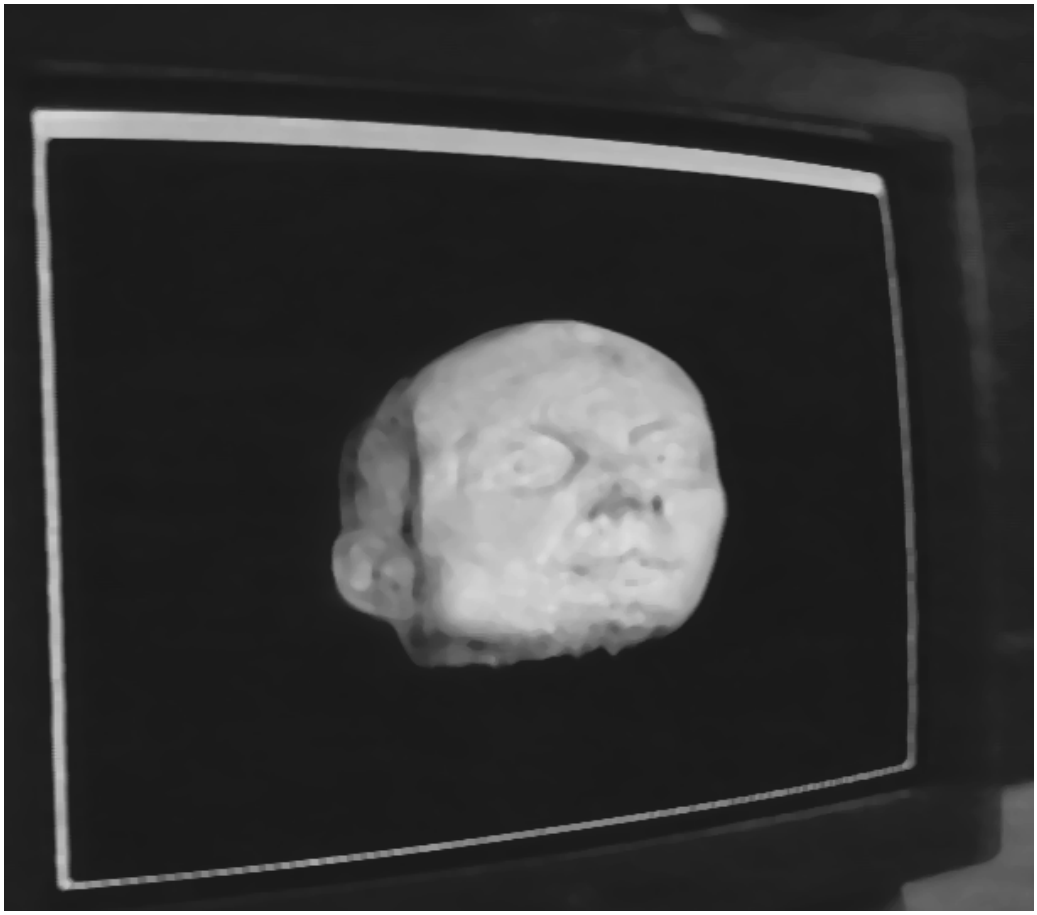

Figura 13: Osservazione di un volume ricostruito da dati CT.

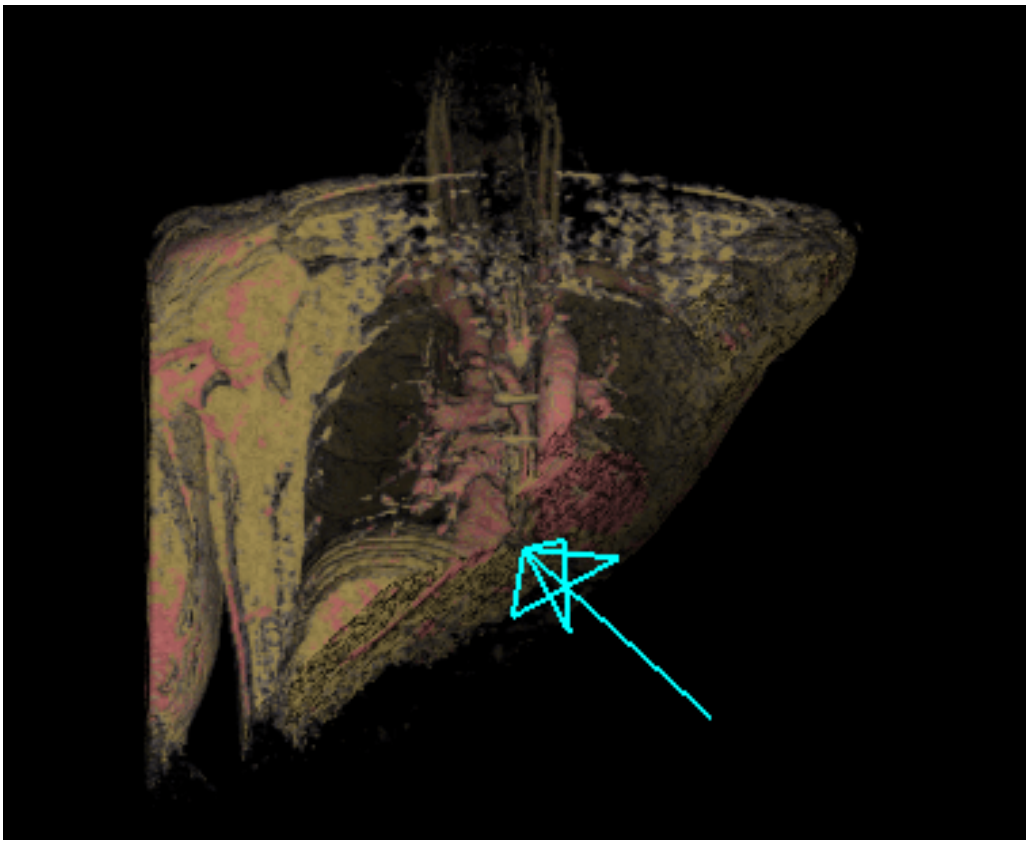

Figura 14: Osservazione e manipolazione di un volume ricostruito da dati MR.

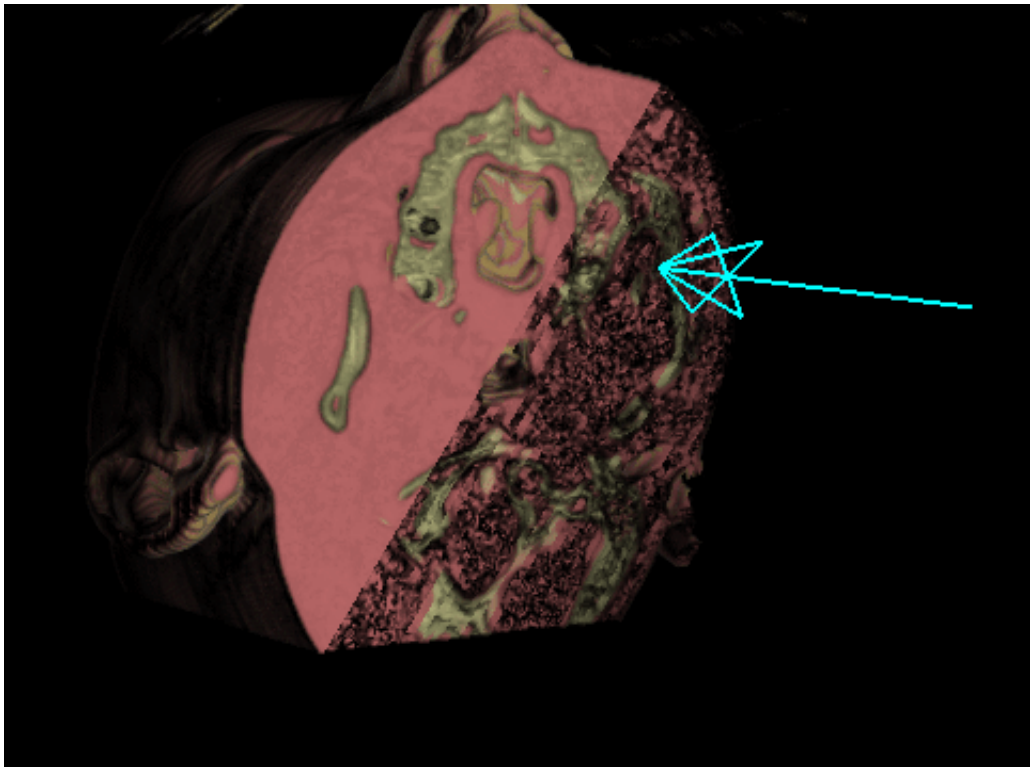

Figura 15: Osservazione e manipolazione di un volume ricostruito da dati CT.

9. Il futuro

Diverse sono le modifiche e le aggiunte che sono in corso di sperimentazione sul TVR.

Le prime sono dirette a conseguire, a parità frequenza di fotogramma, un incremento della qualità della immagine o, viceversa con la stessa qualità, una riduzione dei tempi di rendering.

Fra queste possiamo citare:

- L'escluzione dal volume dei voxels che non danno alcun contributo alla immagine finale. Considerando che spesso l'oggetto in esso racchiuso è circondato da una quantità considerevole di aria che può tranquillamente essere eliminata in quanto di nessun interesse per il medico, si può rappresentare con la stessa risoluzione (stesso numero di voxels) un volume decisamente più contenuto ottenendo una migliore precisione dell'immagine finale.
- L'adozione di un passo variabile nella generazione delle textures. Infatti, dato che la vista è di tipo prospettico, le textures in primo piano appaiono all'osservatore maggiormente distanziate rispetto a quelle più lontane. Addensando opportunamente le textures sulla parte anteriore del volume è possibile far si che, nella immagine finale, queste appaiano equidistanziate. In questo modo si ottiene un incremento della qualità di riproduzione della parte del volume maggiormente visibile a scapito della qualità della parte più nascosta.
- L'adozione di metodi di classificazione più raffinati, come il già citato metodo di Levoy, che consentono una maggiore nitidezza nella riproduzione dei dettagli.
- L'adozione di opportune tecniche di frame buffer per la riduzione del numero dei pixels che ad ogni fotogramma il sistema deve ricalcolare. In particolare, negli spostamenti del volume è possibile incrementare la fluidità di moto aggiornando i pixels della scena alternativamente a scacchiera; si ottiene così un effetto di motion blurr.

Altre modifiche sono dirette ad allargare il campo di utilizzazione del TVR:

- Manipolazione in tempo reale del volume: dall'implementazione di cutting interattivo del volume mediante cutting planes collegati al puntatore di un mouse tridimensionale, all'esecuzione di asportazione di piccole porzioni di volume mediante incisione con strumenti di taglio virtuali, alla deformazione del volume stesso.
- Visualizzazione all'interno del volume di geometrie ottenute mediante segmentazione, per l'evidenziazione di strutture di interesse particolare.
- Combinazione delle precedenti possibilità per la simulazione di particolari interventi chirurgici come l'endoscopia e il posizionamento di stents.

Parallelamente ai precedenti affinamenti della versione *portabile* del TVR, è in corso di realizzazione una nuova versione che utilizza le estensioni per le textures 3D disponibili sulla Silicon. Con questa versione si prevede di ottenere un ulteriore incremento delle prestazioni sia in termini di tempo di elaborazione che di qualità delle immagini. La possibilità di impiegare più tabelle di look-up a diversi livelli rende inoltre plausibile la possibilità di eseguire in tempo reale operazioni quali il normal shading e la classificazione di Levoy.

BIBLIOGRAFIA

(1) Stephen T. Bryson, Sandra Johan: *Time management, simultaneity and time-critical computation in interactive unsteady visualization environments.*
IEEE Visualization, ISBN 0-89791-864-9 (October 1996).

(2) R. Held, N. Durlach: *Telepresence, time delay, and adaptation.*
Pictorial Communication in Real and Virtual Environments. Stephen R . Ellis editor, Taylor and Francis (1991).

(2) G. Mayer, D. Greenberg: *Perceptual color spaces for Computer Graphics.*
Computer Graphics n. 14 1980, pp. 247-261.

(4) B. Cabral, N. Cam, J. Foran: *Accelerated volume rendering and tomographic reconstruction using texture mapping hardware.*
Proceedings 1994 ACM/IEEE Symposium on Volume Visualization, pp. 91-97.

(5) K. Akley: *Reality Engine Graphics.*
Proc. SGGRAPH, pp. 109-116 (1993).

(6) T.J. Cullip, U. Neumann: *Accelerating volume reconstruction with 3D texture hardware.*
Tecnical Report TR93-027, University of North Carolina in Chapel Hill (1993).

(7) O. Wilson, A. Van Gelder, J. Wilhelms: *Direct volume rendering via 3D textures.*
Tecnical Report UCSC-CRL-94-19, University of California in Santa Cruz (1994).

(8) A. Van Gelder, K. Kim: *Direct volume rendering with shading via three-dimensional textures.*
Proc. ACM/IEEE Symposium on Volume Visualization (1996).

(9) R. Turner, E. Gobbetti, I. Soboroff: *Head-tracked stereo viewing with two handed 3D interaction for animated character construction.*
Computer Graphics Forum 15(3), Blackwell. Special issue on Proceedings EUROGRAPHICS Conference, Poitiers, France (1996).

(10) J. Roese, L. Mc Cleary: *Stereoscopic computer graphics for simulation and modeling.*
Proc. SIGGRAPH, pp. 41-47 (1979).

(11) E. Hibbard: *On the theory and applications of stereographics in scientific visualization.*
Eurographics state of the art Reports, pp. 1-21 (1991).

(12) L. Hodges: Basic principles of stereographics software development.
Proc. SPIE (1991).

SOMMARIO

INDICE DELLE FIGURE

(1) Stephen T. Bryson, Sandra Johan: *Time management, simultaneity and time-critical computation in interactive unsteady visualization environments.*
IEEE Visualization, ISBN 0-89791-864-9 (October 1996).

(2) R. Held, N. Durlach: *Telepresence, time delay, and adaptation.*
Pictorial Communication in Real and Virtual Environments. Stephen R. Ellis editor, Taylor and Francis (1991).

(3) G. Mayer, D. Greenberg: *Perceptual color spaces for Computer Graphics.*
Computer Graphics n. 14 1980, pp. 247-261.

(4) B. Cabral, N. Cam, J. Foran: *Accelerated volume rendering and tomographic reconstruction using texture mapping hardware.*
Proceedings 1994 ACM/IEEE Symposium on Volume Visualization, pp. 91-97.

(5) K. Akley: *Reality Engine Graphics.*
Proc. SGGRAPH, pp. 109-116 (1993).

(6) T.J. Cullip, U. Neumann: *Accelerating volume reconstruction with 3D texture hardware.*
Tecnical Report TR93-027, University of North Carolina in Chapel Hill (1993).

(7) O. Wilson, A. Van Gelder, J. Wilhelms: *Direct volume rendering via 3D textures.*
Tecnical Report UCSC-CRL-94-19, University of California in Santa Cruz (1994).

(8) A. Van Gelder, K. Kim: *Direct volume rendering with shading via three-dimensional textures.*
Proc. ACM/IEEE Symposium on Volume Visualization (1996).

(9) R. Turner, E. Gobbetti, I. Soboroff: *Head-tracked stereo viewing with two handed 3D interaction for animated character construction.*
Computer Graphics Forum 15(3), Blackwell. Special issue on Proceedings EUROGRAPHICS Conference, Poitiers, France (1996).

(10) J. Roese, L. Mc Cleary: *Stereoscopic computer graphics for simulation and modeling.*
Proc. SIGGRAPH, pp. 41-47 (1979).

(11) E. Hibbard: *On the theory and applications of stereographics in scientific visualization.*
Eurographics state of the art Reports, pp. 1-21 (1991).

(12) L. Hodges: Basic principles of stereographics software development.
Proc. SPIE (1991).