



UNIVERSITÀ DEGLI STUDI DI SIENA

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica

Rendering visio-aptico
di volumi ecografici ed applicazioni
alla ginecologia ed ostetricia

Relatore

Prof. Ing. Domenico Prattichizzo

Correlatori

Prof. Ing. Antonio Vicino

Prof. Dott. Felice Petraglia

Dott. Filiberto Maria Severi

Tesi di laurea di

Berardino la Torre

Anno Accademico 2001-2002

Indice

1	Medical Imaging	5
1.1	Acquisizione delle immagini	5
1.1.1	Sistema ecografico	5
1.1.2	DICOM	12
1.2	Filtraggio	17
1.2.1	Gaussian Smoothing	17
1.2.2	Caso monodimensionale (1D)	17
1.2.3	Caso bidimensionale (2D)	19
1.2.4	Caso tridimensionale (3D)	20
1.3	Segmentazione	22
1.3.1	Thresholding	23
1.4	Ricostruzione e visualizzazione 3D	25
1.4.1	Marching squares	28
1.4.2	Marching cubes	32
2	Interfacce aptiche	36
2.1	Psicofisica	36
2.2	Note storiche	37
2.3	Il PHANToM	38
2.4	Collision detection	40
2.4.1	OBB-Tree	42

2.4.2	Usare l'OBBTTree per la collision detection	45
2.5	Proxy	47
2.6	Force rendering	48
3	Utilizzo delle interfacce aptiche in medicina	52
4	Progettazione <i>Software</i>	56
4.1	<i>US3D</i>	57
4.2	<i>US3DTouch</i>	64
5	Librerie	69
5.1	FLTK	69
5.2	VTK	73
5.2.1	Introduzione	73
5.2.2	Graphics model	73
5.2.3	Visualization model	74
5.2.4	Data object	77
5.2.5	Tipi di celle	78
5.2.6	Tipi di DataSet	79
5.3	GHOST SDK	83
A	CD-ROM	88
A.1	Contenuto	88
A.2	Compilazione del codice	89
B	Articolo	92
	Bibliografia	93

Elenco delle figure

1.1	Sonda ecografica	6
1.2	Fasci di ultrasuoni	7
1.3	Scrittura degli echi di ritorno	7
1.4	Esempio d'indagine ecografica	8
1.5	Ipotesi di parallelità tra i piani immagine	8
1.6	Tipi di scansione	9
1.7	Siemens Sonoline Elegra	10
1.8	Modalità di acquisizione : Lineare (a) e (b). Rocked (c)	11
1.9	DICOM standard stack	13
1.10	Struttura di un file in formato DICOM	14
1.11	Header di una file DICOM	15
1.12	DICOM File Meta Information. Struttura di una etichetta (tag)	16
1.13	Immagini memorizzate nel DICOM Data Set	16
1.14	distribuzione gaussiana monodimensionale (a) continua. (b) discretizzata.	18
1.15	Kernel 1D	18
1.16	set di dati. (a) originale. (b) filtrato.	18
1.17	Distribuzione gaussiana bidimensionale (a) continua. (b) discre- tizzata.	19
1.18	Kernel bidimensionale	19
1.19	Esempio di immagine ecografica filtrata $\sigma = 5$	20

1.20 Sequenza di immagini	20
1.21 Kernel 3D	21
1.22 Esempio di una immagine composta da tre zone omogenee	24
1.23 Esempio di estrazione di una regione di interesse	24
1.24 Immagine binaria	25
1.25 Modalità di resa tridimensionale di un set di dati volumetrico	26
1.26 a : dati volumetrici; b : piano immagine; c : fascio di raggi	27
1.27 Esempio di rendering volumetrico diretto utilizzando diverse funzioni di blending.	27
1.28 Immagine a livelli di grigio. In (a) sono riportati i valori di intensità di grigio. (b) $P(i, j)$ è l'intensità di grigio associato al pixel alla i -esima riga e j -esima colonna	28
1.29 Esempio di isolinea definita sull'intervallo $[\mathbf{T}_L, \mathbf{T}_H] = [231, 247]$	29
1.30 Posizionamento della griglia sull'immagine di Fig 1.28	30
1.31 Classificazione dei vertici di ciascun quadrato $S(i, j)$ ottenuta usando come intervallo di soglia $[\mathbf{T}_L, \mathbf{T}_H] = [231, 247]$	30
1.32 Configurazioni possibili per ciascun quadrato	31
1.33 Calcolo del segmento di isolinea relativo al quadrato $S(i,j)$	31
1.34 Isolinea cercata	31
1.35 Classificazione dei voxel	32
1.36 (a) classificazione dei vertici del cubo $C(i, j, k)$. (b) contributo di isosuperficie generata da $C(i,j,k)$	33
1.37 Tabella di classificazione dei cubi	34
1.38 Esempio di isosuperficie estratta da un set di dati volumetrico. Si puo osservare come incide sulla superficie il processo di decimazione e quello di smoothing. (a) 66872 triangoli, (b)decimazione 50% : 31874 triangoli, (c) decimazione 90% : 5610 triangoli, (e) caso (c) con smoothing 50%	35

2.1	PHANToM	38
2.2	Scena virtuale composta da due oggetti, uno statico O_B ed uno dinamico O_A che va a collidere al tempo t_2 con O_B	40
2.3	Collision detection. H_P : Posizione dell'interfaccia aptica	41
2.4	Processo aptico	41
2.5	OBBTree	42
2.6	Calcolo di un OBB	44
2.7	Particolare di tre OBB che tassellano una superficie sferica	45
2.8	Test di collision detection : (a) Nessuna collisione. (b) Collisione solo con l'OBBTree. (c) Collisione con l'oggetto VO (Virtual Object).	46
2.9	Calcolo del proxy	49
2.10	Esempio di oggetto cedevole	50
2.11	Modello fisico del sistema usato per il processo di force rendering.	51
3.1	Sistema chirurgico robotizzato della Intuitive Surgical.	53
3.2	Strumentazione chirurgica robotizzata: (a) ditali attuati con feedback di forza. (b) microbisturi che operano in via laparoscopica	54
4.1	Interfaccia utente. In alto a destra si puo vedere la visualizzazione volumetrica diretta	57
4.2	Pannello dati	58
4.3	Sistemi di riferimento dei tre piani di vista axial, sagittal, coronal.	59
4.4	Pannello di vista e di filtraggio del volume.	60
4.5	(a) Estrazione del volume di interesse. (b) Identificazione della isosuperficie di interesse.	61
4.6	Tipi di ombreggiatura usati nel processo di surface rendering: (a) flat shading. (b) gouraund shading	62
4.7	Esempio di surface rendering.	62

4.8	Pipeline di ricostruzione 3D	63
4.9	Interfaccia utente	64
4.10	(a) OBB-Tree livello 0. (b) OBB-Tree livello 9. (c) Effetto di trasparenza, si vedono il proxy e la posizione del PHANToM. (d) Struttura della superficie, 16.702 triangoli.	66
4.11	Workstation visio-aptica in funzione presso il laboratorio di Automatica e Robotica del Dipartimento di Ingegneria dell'Informazione dell'Università di Siena.	67
4.12	Struttura di US3DTouch ed interfacciamento con il PHANToM	68
5.1	Esempio FLTK : Hello World	71
5.2	Organizzazione di FLUID	72
5.3	Esempio FLUID : Hello World	72
5.4	Esempio di definizione di una cella	79
5.5	Tipi di celle	80
5.6	Tipi di dataset	82
5.7	Esempio di definizione di un'oggetto	84
5.8	Struttura gerarchica della scena	85
A.1	(a) settaggio dei path dei file di include (.h). (b) settaggio dei path dei file di library (.lib).	90

Ringrazio il Prof. Domenico Prattichizzo, il Prof. Antonio Vicino e Federico Barbagli per il continuo incoraggiamento e supporto allo sviluppo della tesi, ed il Dipartimento di Pediatria, Ostetricia e Medicina della Riproduzione nella persona del Dott. Filiberto Maria Severi e del Prof. Dott. Felice Petraglia.

"dedicata a nonno Berardino..."

Introduzione

Lo sviluppo di sistemi medicali sempre più avanzati per l’acquisizione delle immagini finalizzati all’estrazione di informazioni tridimensionali sugli organi del corpo umano tramite metodi non invasivi, hanno portato all’affermazione di una nuova disciplina scientifica nota come *medical imaging*. Nell’ambito di questa disciplina rivestono un ruolo fondamentale le conoscenze acquisite in campi tra loro diversi e complementari, come l’elaborazione delle immagini e la visualizzazione scientifica. La possibilità offerta dallo sviluppo delle tecnologie informatiche e robotiche consente di realizzare strumenti basati su soluzioni tecnologicamente sempre più avanzate.

Fra gli strumenti diagnostici più importanti, si annoverano la Tomografia Assiale Computerizzata (TAC) e la Risonanza Magnetica Nucleare (MRI). Tali sistemi sono caratterizzati da una elevata accuratezza sia nell’acquisizione delle immagini e sia nell’allineamento assiale delle stesse.

In questa tesi si sono studiate le tecniche di medical imaging applicate alla diagnostica ultrasonica, che risulta essere innocua per la salute del paziente, meno costosa, ma caratterizzata da una bassa accuratezza del sistema di acquisizione. Nell’ultrasonografia la sonda di acquisizione non è vincolata a guide meccaniche ma segue il libero movimento della mano dell’operatore.

La prima parte della tesi è dedicata alla progettazione di un software per l’estrazione di un modello geometrico tridimensionale del volume ecografico in esame. Gli elementi principali del processo di ricostruzione sono quelli propri

del *medical imaging* (filtraggio, segmentazione, estrazione della regione di interesse, estrazione di isosuperfici). L'ipotesi di lavoro è che la scansione ecografica sia tale da generare una sequenza di immagini 'quasi' parallele. Lo strumento impiegato per l'acquisizione dei dati ecografici è il *Siemens Sonoline Elegria Millennium Edition* disponibile presso il Dipartimento di Pediatria, Ostetricia e Medicina della Riproduzione dell'Università di Siena.

Il secondo obiettivo di questa tesi è quello di integrare le tecnologie di medical imaging con quelle aptiche per il rendering tattile di volumi virtuali. Il fine ultimo è quello di offrire all'utente la possibilità di un'interazione multi sensoriale con il modello tridimensionale ricostruito. Le interfacce aptiche sono strumenti in grado di restituire un feedback di forza all'utente che interagisce con un ambiente virtuale, simulando così la sensazione tattile.

Le problematiche affrontate sono molteplici e vanno dalla *collision detection* al *force rendering*. Il lavoro di tesi si è quindi concretizzato nella progettazione e realizzazione di una *workstation* visio-aptica basata sull'interfaccia PHANTOM della *SensAble Technologies Inc.*

Si sono studiate alcune applicazioni di interesse nel campo ostetrico e ginecologico, come l'interazione visio-aptica con i modelli virtuali di un feto e di cisti ovariche costruiti a partire da immagini ecografiche. Si stanno attualmente analizzando le possibili applicazioni di questo sistema per il *training* di personale medico specializzato.

Capitolo 1

Medical Imaging

In questo capitolo verranno introdotti gli strumenti propri del medical imaging attraverso cui è possibile, a partire da un set di dati volumetrico (come una sequenza di immagini generata da una scansione ecografica), estrarre informazioni di tipo geometrico, come la ricostruzione tridimensionale di un elemento d'interesse presente nel volume. L'estrazione delle isosuperfici è l'elemento centrale della ricostruzione tridimensionale.

1.1 Acquisizione delle immagini

1.1.1 Sistema ecografico

Le tecniche di diagnostica ecografiche utilizzano gli ultrasuoni per ricostruire immagini di organi interni [3]. Gli ultrasuoni (US) sono così denominati poichè indicano frequenze sonore poste al di sopra di quelle normalmente percepite dall'orecchio umano (16-20.000 Hz). In ecografia si utilizzano frequenze variabili da 2MHz a 20MHz. Ogni apparecchio utilizzato per eseguire un esame ecografico (Ecografo o Ecotomografo) è dotato di una sonda Fig 1.1 che emette fasci di ultrasuoni diretti all'interno del corpo (Fig 1.2-A), le onde riflesse (echi

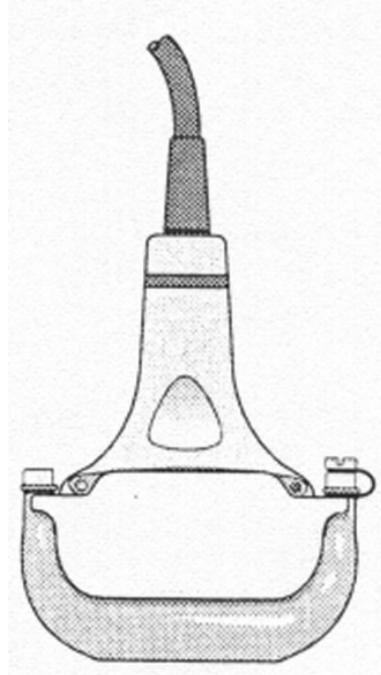


Figura 1.1: Sonda ecografica

di ritorno) (Fig 1.2-B) dai diversi tessuti sono poi captate da un trasduttore posto sulla sonda e trasformate in array di dati bidimensionali [16].

Esistono diverse modalità di scrittura degli echi di ritorno (Fig 1.3). Un tracciato molto diffuso in cardiologia è rappresentato dal *TM-mode* utilizzato soprattutto per analizzare i movimenti delle valvole cardiache. La rappresentazione dei suoni più utilizzata è sicuramente l'immagine *B-mode*¹, dove ad ogni eco corrisponde un punto luminoso. In questa tesi si considerano immagini ecografiche del tipo *B-mode*. Attraverso l'elaborazione di tutti i punti si ottiene una riproduzione bidimensionale (immagine) che rappresenta una sezione del volume sotto indagine (Fig 1.4). Ogni pixel dell'immagine rappresenta un'unità elementare del volume totale, tale unità prende il nome di *voxel*² [14].

L'ecografia rappresenta oggi il test di screening ideale in molteplici indagini

¹B indica la luminosità (Brightness)

²Volume Element [mm^3]

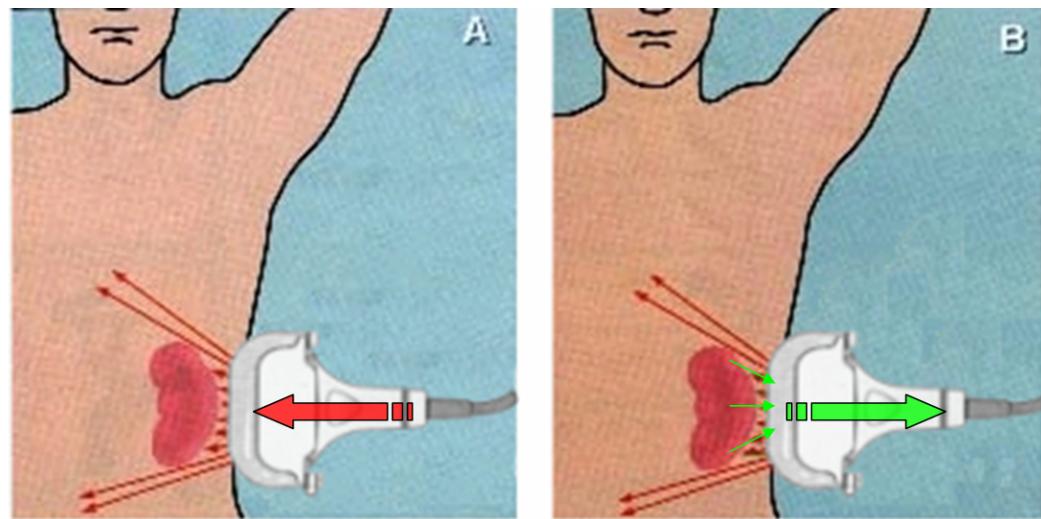


Figura 1.2: Fasci di ultrasuoni

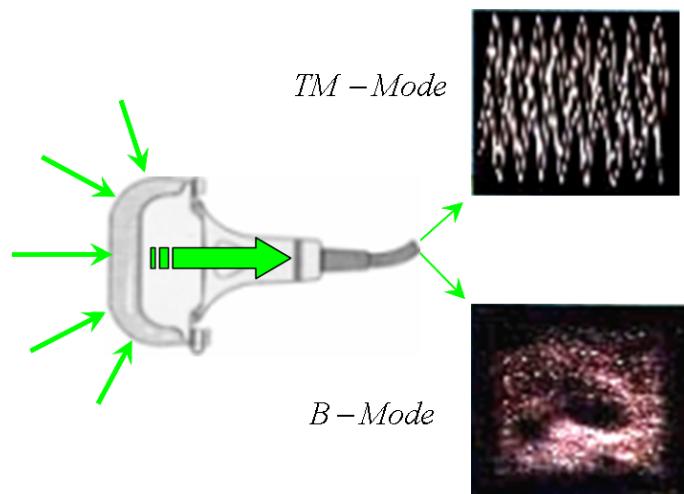


Figura 1.3: Scrittura degli echi di ritorno



Figura 1.4: Esempio d'indagine ecografica

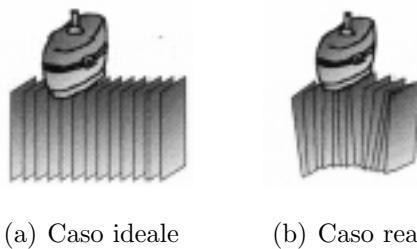


Figura 1.5: Ipotesi di parallelità tra i piani immagine

epidemiologiche e di medicina preventiva, questo è dovuto alla non pericolosità dell'esame, al costo limitato dell'apparecchiatura, se paragonato ad altri sistemi (TAC,MRI) ed alla possibilità di ottenere immagini in tempo reale.

Il fatto però che la sonda non sia vincolata a delle guide ma segue il movimento libero della mano dell'operatore, fa sì che i piani di sezione del volume, che ciascuna immagine rappresenta, possano avere orientamento diversi l'uno dall'altro, Fig 1.5 (b). Si vedrà più avanti che per poter applicare l'algoritmo di ricostruzione tridimensionale, descritto in questa tesi, sarà, necessario assumere, che i piani di sezione siano allineati.

L'ipotesi di base della tesi è quindi che l'operatore esegua una scansione ecografica che garantisca, per quanto possibile, l'allineamento assiale dei piani immagine, Fig 1.5 (a).

I dati elaborati in questa tesi sono stati acquisiti con un ecografo avanzato, Siemens Sonoline Elegra (Fig 1.7), che dispone di alcuni strumenti di ausilio

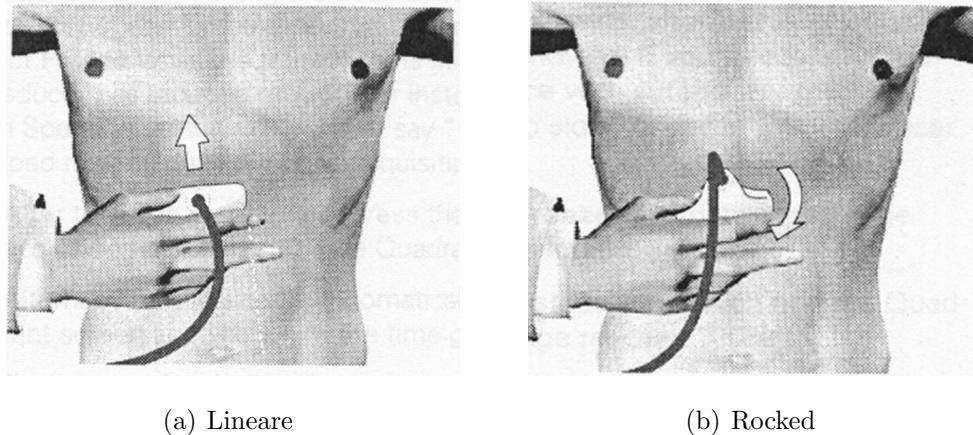


Figura 1.6: Tipi di scansione

all'operatore nella fase di scansione. In particolare è possibile selezionare due diverse modalità di acquisizione:

- Lineare, Fig 1.6 (a)
- Rocked, Fig 1.6 (b)

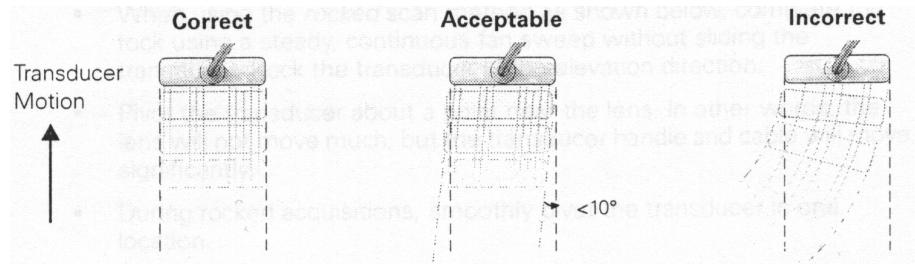
Quando viene usata la scansione *Lineare* la sonda deve essere posizionata perpendicolarmente alla pelle e la traiettoria seguita deve essere il più possibile stabile e regolare. Un'acquisizione troppo veloce comporta la comparsa di bande nere nel volume finale e una distorsione della geometria delle immagini [30].

I modi di operare per ottenere una corretta acquisizione sono riportati in Fig 1.8(a) e Fig 1.8(b) per la modalità di scansione *Lineare* e in Fig 1.8(c) per la modalità di scansione *Rocked*, dove è assunto che l'operatore esegua l'acquisizione ruotando la sonda intorno ad un punto fisso.

In questa tesi si farà riferimento alla scansione *Lineare* che consente di acquisire immagini su piani allineati.

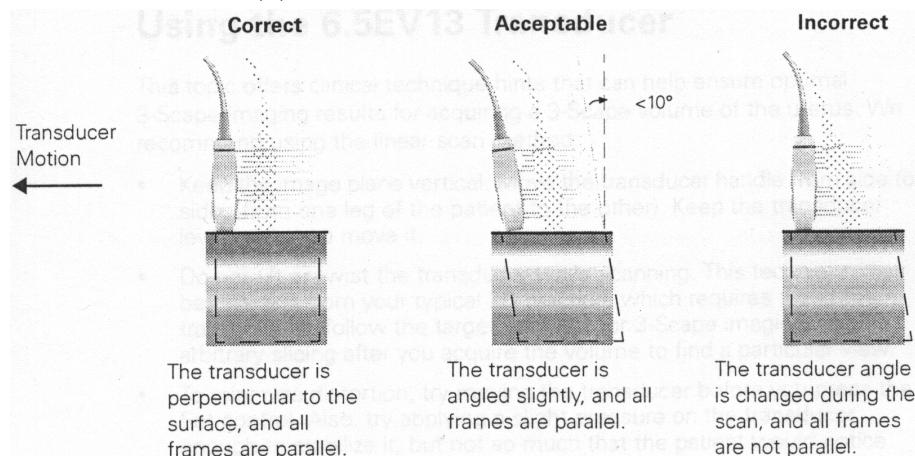


Figura 1.7: Siemens Sonoline Elegra

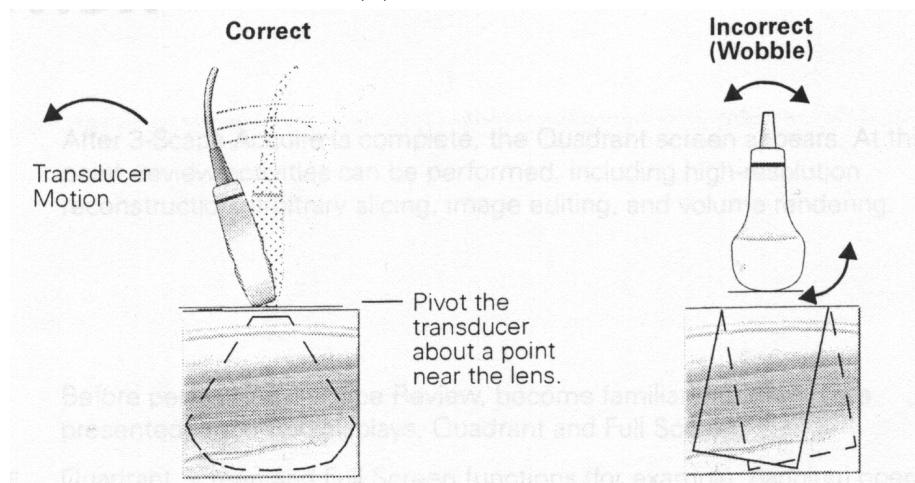


The transducer is moved straight, and all image frames are kept parallel.
The transducer is moved at a small angle ($<10^\circ$), and all image frames are kept parallel.
The transducer is moved at an angle, and all frames are not parallel.

(a) Posizionamento della sonda



(b) Acquisizione



(c) Acquisizione

Figura 1.8: Modalità di acquisizione : Lineare (a) e (b). Rocked (c)

1.1.2 DICOM

L'introduzione della tomografia computerizzata (CT) seguita da altri sistemi diagnostici digitali ed il sempre maggiore utilizzo dei computer in applicazioni cliniche, ha reso necessario la definizione di standard per lo scambio di informazioni tra sistemi medicali di diversi produttori. Nel 1983 l' ACR³ e il NEMA⁴ hanno definito lo standard **DICOM**⁵ [17]. Lo stack dello standard **DICOM** è riportato in Fig 1.9 e le sue funzioni sono quelle di:

- promuovere la comunicazione di informazioni tra dispositivi di diversi produttori.
- facilitare lo sviluppo e l'espansione dei sistemi di comunicazione e di archiviazione delle immagini (PACS⁶) con la possibilità di interfacciare sistemi informativi di diversi ospedali.
- favorire lo sviluppo di un sistema informativo diagnostico con la possibilità di essere interrogato da una varietà di dispositivi distribuiti geograficamente.

Lo standard DICOM giunto alla versione 3 è a tutti gli effetti un protocollo di rete che fornisce specifiche per i diversi livelli di comunicazione. Prevede le facilities degli standard di rete per l'interconnessione (TCP-IP e ISO-OSI), la negoziazione dei messaggi, la specificazione object-oriented di set di dati e classi di servizi.

Il supporto dello standard DICOM da parte del sistema Siemens Sonoline Elegra, permette, una volta eseguita la scansione ecografica, di salvare i dati in un file in formato **DICOM** (Fig 1.10) su disco ottico.

³American College of Radiology

⁴National Electrical Manufacturers Association

⁵Digital Imaging and Communications in Medicine

⁶Picture Archiving and Communication Systems

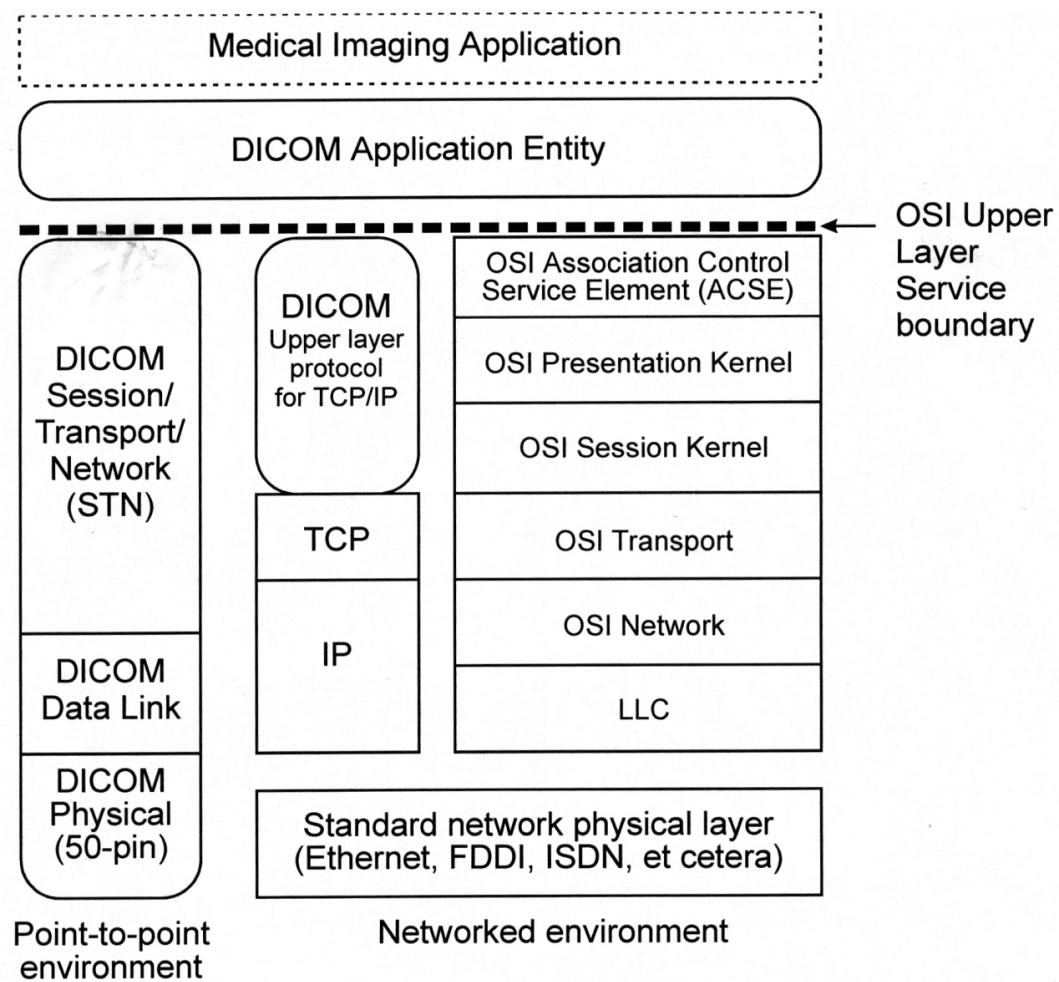


Figura 1.9: DICOM standard stack



Figura 1.10: Struttura di un file in formato DICOM

Un file in formato **DICOM**, Fig 1.10, è costituito da un header ASCII, riportato in Fig 1.11, strutturato ad etichette (tag Fig 1.12) con campi di lunghezza variabile, contenente informazioni sul paziente e sulle modalità di acquisizione, seguito dal set di dati, che nel caso di un sistema ecografico è costituito da una sequenza di immagini memorizzate per righe e non compresse, con una risoluzione di 256 livelli di grigio (Fig 1.13).

Analizzando nel dettaglio l'header di Fig 1.11, si possono trovare tutte le informazioni che caratterizzano sia il processo di acquisizione che il paziente sotto esame. Da notare il tag *motionMode* che identifica il tipo di scansione, che per quanto detto prima, (Pag 9), deve essere di tipo lineare, mentre il tag *mode* è B e rappresenta il tipo di visualizzazione degli echi (B-Mode), *voxelWidth*, *voxelHeight*, *voxelDepth* indicano le dimensioni spaziali dell'unità di volume (voxel). Infine *volumeWidth*, *volumeHeight* sono le dimensioni di ciascuna immagine e *volumeDepth* indica il numero di immagini acquisite. Le informazioni sul paziente e sull'esame eseguito sono infine organizzate nei due record *patientRecord* e *studyRecord*.

```

3DVolume {
    version 2
    UID "1.3.12.2.1107.5.5.7414.11107.164919.105"
    AcquisitionSequenceReferenceUID = "1.3.12.2.1107.5.5.7414.11107.164919.106"
    instanceDateTime 3182604585
    label "xxxx"
    patientRecord {
        version 1
        lastName "Unknown"
        firstName ""
        middleName ""
        ID "0111071633"
        birthMonth 0
        birthDay 0
        birthYear 0
        weight 0.000000
        height 0.000000
        gender OTHER
    }
    studyRecord {
        version 2
        UID "1.3.12.2.1107.5.5.7414.11107.161754.93"
        instanceDateTime 3182604136
        description "Unknown"
        referringPhysician ""
        refPhyPre ""
        refPhyFirst ""
        refPhyMiddle ""
        refPhyLast ""
        refPhySuf ""
        accessionNumber ""
        admittingDiagnosis ""
    }
    seriesRecord {
        version 3
        UID "1.3.12.2.1107.5.5.7414.11107.161755.95"
        instanceDateTime 3182604136
        modality US
        operatorName ""
        imageNumber 4
    }
    3DVolumeData {
        version 1
        voxelWidth 0.986291
        voxelHeight 0.986291
        voxelDepth 0.986291
        3DAcquisitionInfo {
            version 3
            mode B
            motionMode LINEAR
            probeType 35C40
            probeExamId 10
            butterflyState 6
            qualFactor 4
        }
        bData {
            version 2
            volumeWidth 237
            volumeHeight 174
            volumeDepth 165
            data 0x700
        }
    }
}

```

Figura 1.11: Header di una file DICOM

Attribute Name	Tag	Type	Attribute Description
----------------	-----	------	-----------------------

Figura 1.12: DICOM File Meta Information. Struttura di una etichetta (tag).

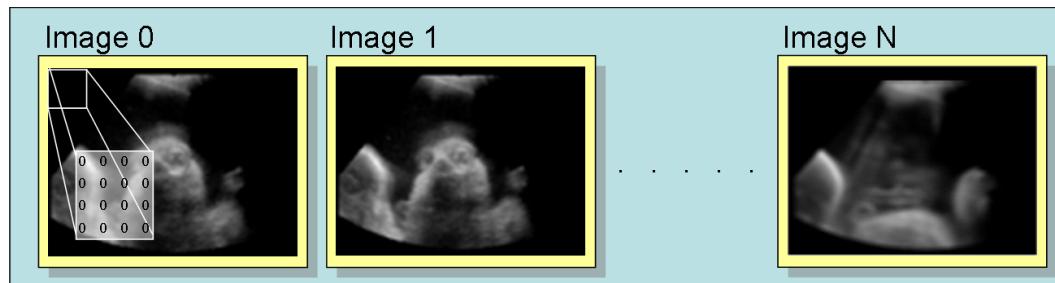


Figura 1.13: Immagini memorizzate nel DICOM Data Set

1.2 Filtraggio

Tutti i processi di acquisizione di immagini digitali, sono caratterizzati da una componente di rumore indotta dal sistema di acquisizione, che degrada la qualità e quindi il contenuto informativo dei dati [16]. Pertanto è sempre necessaria una operazione di filtraggio (post-acquisizione) atta ad eliminare, per quanto possibile, tali disturbi. Le possibilità offerte in tal senso dall'*elaborazione delle immagini* sono molteplici. In questa tesi si è considerato un particolare filtro passa basso descritto di seguito.

1.2.1 Gaussian Smoothing

Il filtro di smoothing gaussiano è un operatore di convoluzione, usato per sfumare l'immagine e ridurne il rumore. In tal senso è simile ad un *filtro mediano*,⁷ con la differenza che viene utilizzato un *kernel*⁸ di convoluzione ottenuto da una gaussiana.

1.2.2 Caso monodimensionale (1D)

La distribuzione gaussiana monodimensionale (Fig 1.14 (a)) è definita da :

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad (1.1)$$

dove σ è la deviazione standard, ed è stato assunto il valore medio nullo. Dato un set di dati, come quello in Fig 1.16 (a), e discretizzando la distribuzione gaussiana come in Fig 1.14 (b), si ottiene il kernel del filtro, Fig 1.15, che convoluto con il set di dati produce il risultato di Fig 1.16 (b). Si osserva

⁷Il filtro mediano è un tipo di filtraggio semplice, intuitivo e facile da implementare. L'idea è quella di associare a ciascun pixel il valore medio calcolato sui pixel adiacenti

⁸Il kernel usato per l'operazione di filtraggio è solitamente una matrice 3x3,5x5, di numeri usata come maschera di convoluzione con l'immagine

come le brusche variazioni (quindi le componenti in alta frequenza) sono state eliminate dal filtraggio passa basso, rendendo così i dati in uscita più regolari.

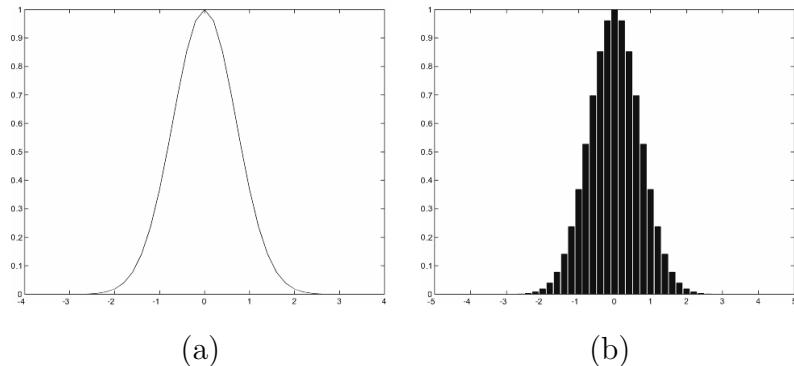


Figura 1.14: distribuzione gaussiana monodimensionale (a) continua. (b) discretizzata.



Figura 1.15: Kernel 1D

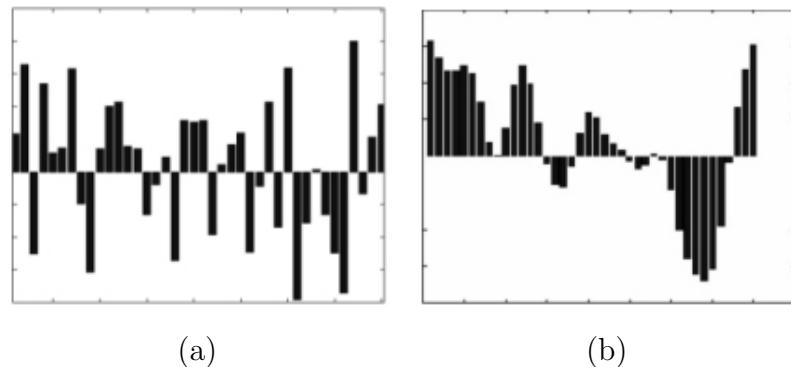


Figura 1.16: set di dati. (a) originale. (b) filtrato.

1.2.3 Caso bidimensionale (2D)

Nel caso di set di dati bidimensionali, la gaussiana diventa (Fig 1.17 (a)) :

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1.2)$$

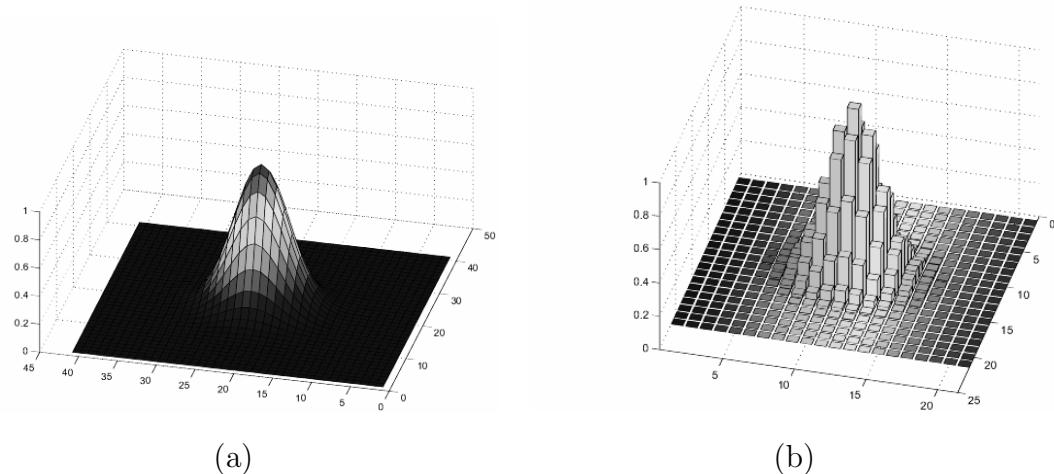


Figura 1.17: Distribuzione gaussiana bidimensionale (a) continua. (b) discretizzata.

Come nel caso monodimensionale, è necessario discretizzare tale distribuzione Fig 1.17 (b), per ottenere il *kernel* Fig 1.18 da convolvere, con il set di dati bidimensionale (immagine), riportato in Fig 1.19.

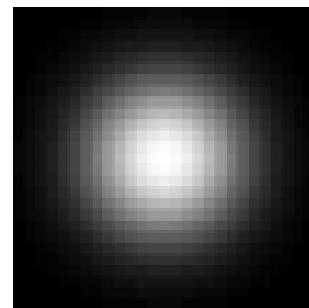


Figura 1.18: Kernel bidimensionale

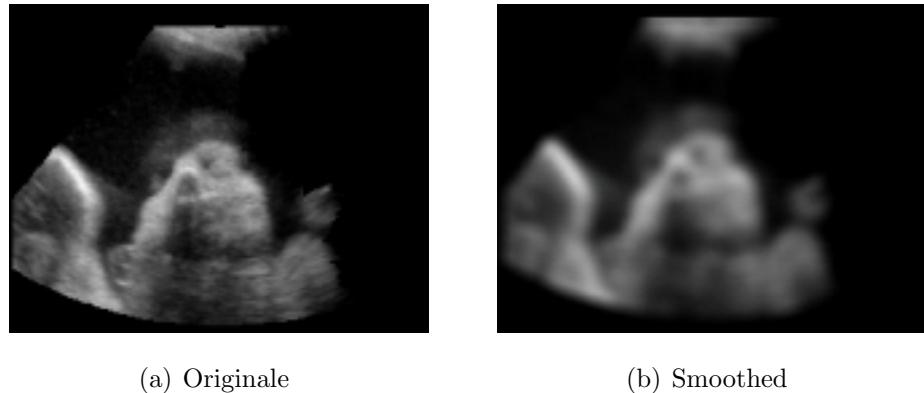


Figura 1.19: Esempio di immagine ecografica filtrata $\sigma = 5$

1.2.4 Caso tridimensionale (3D)

Nel caso di set di dati tridimensionali, come quello che si ottiene con una sequenza di immagini ecografiche (Fig 1.20), è possibile eseguire l'operazione di filtraggio:

- usando un kernel bidimensionale su tutte le immagini
- usando un kernel tridimensionale così da utilizzare anche l'informazione contenuta nelle immagini precedenti e successive, a quella da filtrare.

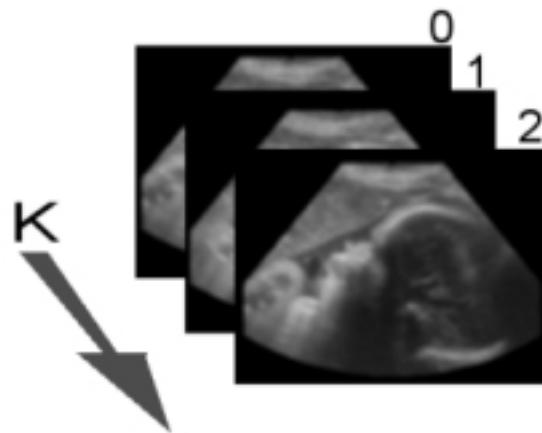


Figura 1.20: Sequenza di immagini

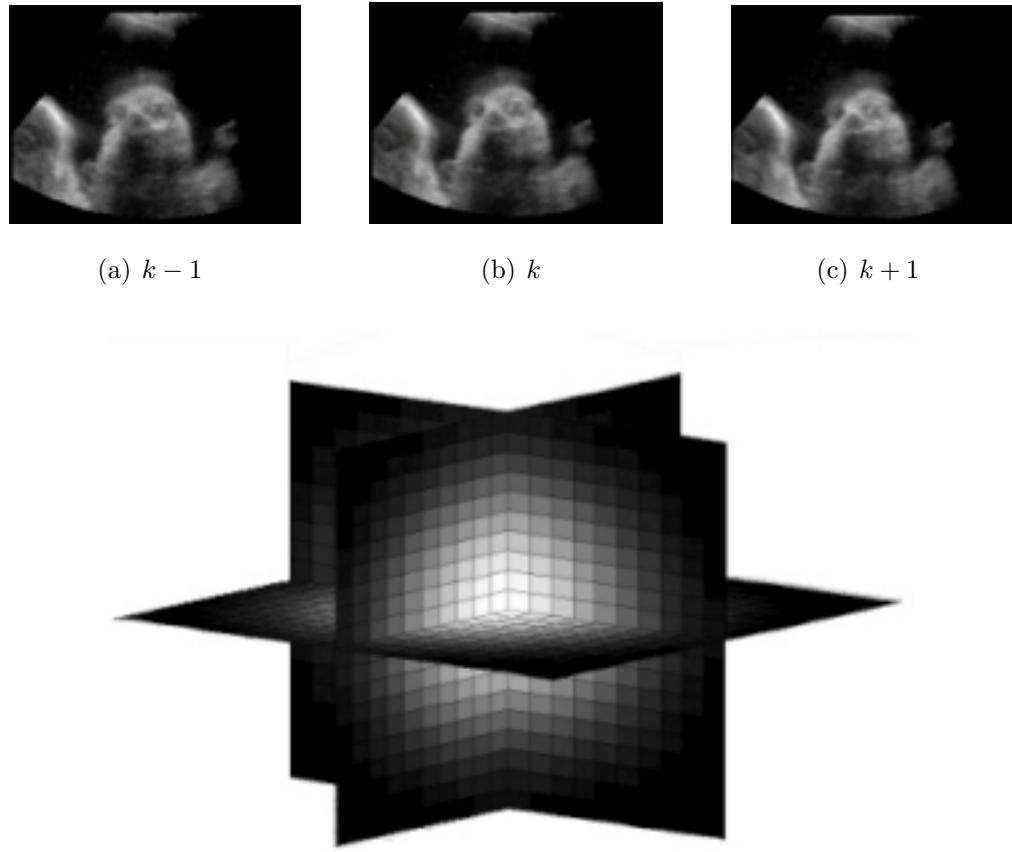


Figura 1.21: Kernel 3D

Il primo metodo è da usare con sequenze di immagini scorrelate. Al contrario in una scansione ecografica si vede come l'immagine k sia strettamente correlata alle immagini $(k - 1)$ e $(k + 1)$.

Di seguito è riportata l'espressione che definisce una gaussiana in tre dimensioni, le cui superfici di livello costituiscono delle sfere, che discretizzate portano al *kernel* del filtro riportato in Fig 1.21.

$$G(x, y, z) = \frac{1}{\sqrt{8\pi^3}\sigma^3} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad (1.3)$$

1.3 Segmentazione

E' il processo attraverso cui un'immagine medica viene suddivisa in regioni che identificano le diverse strutture anatomiche. Le tecniche di segmentazione possono essere divise in tre categorie :

- Manuali
- Semi automatiche
- Automatiche

La prima categoria include tutte quelle tecniche, dove l'operatore non è assistito, in alcun modo, durante il processo di segmentazione. Ma è lui stesso che deve marcare le differenti regioni con il mouse o altri dispositivi di puntamento.

La seconda categoria contempla, invece, tutti quegli algoritmi che aiutano l'operatore durante il processo di segmentazione. Per esempio, l'operatore individua ed identifica dei punti significativi nell'immagine che sa appartenere a regioni differenti, è poi il processo di segmentazione che in modo automatico cerca di trovare nuovi punti, partendo da quelli selezionati manualmente.

Della terza categoria fanno invece parte tutte quelle tecniche che non richiedono un particolare input da parte dell'operatore. Un processo di segmentazione totalmente automatico generalmente deve avere delle conoscenze circa le strutture presenti nell'immagine⁹, questo perchè un computer non ha "*alcuna informazione*" su quello che "*vede*". Un criterio per eseguire un processo di segmentazione automatico, è quello che si basa sulla selezione dei livelli di grigio (thresholding) anzichè sulla topologia o la disposizione spaziale dei punti.

⁹per esempio conoscenze di tipo geometrico o fisiche

1.3.1 Thresholding

Una semplice implementazione di tale tecnica prevede in uscita un’immagine binaria¹⁰. I pixel di colore nero, rappresentano lo sfondo mentre quelli di colore bianco, rappresentano l’oggetto di interesse (o viceversa). Il thresholding può essere determinato da uno o più parametri. Se ne viene utilizzato uno solo questo prende il nome di *intensity threshold* (\mathbf{T}) ed ogni pixel dell’immagine viene comparato con questa soglia: se l’intensità del pixel è maggiore¹¹ allora il pixel viene settato ad 1. Altrimenti è settato a 0.

Un altro criterio, invece, è quello di definire una banda di intensità $[\mathbf{T}_L, \mathbf{T}_H]$: se l’intensità del pixel è interno¹² a tale intervallo allora viene settato a 1 altrimenti a 0.

Si consideri l’immagine in Fig 1.22. Dall’istogramma è possibile riconoscere tre zone in cui vi è una concentrazione di pixel. Questo vuol dire che nell’immagine vi è la presenza di tre zone quasi omogenee, che possiamo individuare come:

- (a): struttura ossea
- (b): cervello
- (c): sfondo

infatti se si estraе dall’immagine la parte relativa alla banda (a) dell’istogramma, quello che si ottiene è l’immagine di Fig 1.23. Quindi ciò vuol dire che eseguendo una operazione di thresholding sull’immagine di partenza (Fig 1.22 (a)) utilizzando come intervallo di soglia $[\mathbf{T}_L, \mathbf{T}_H]$ la banda *a* quello che si ottiene è l’immagine binaria di Fig 1.24.

¹⁰un’immagine binaria è definita su due livelli di grigio nero o bianco. Viene utilizzato un solo bit per rappresentarla 0=nero 1=bianco

¹¹oppure minore, dipende dal criterio di thresholding adottato

¹²oppure, esterno, dipende sempre dal criterio di thresholding adottato

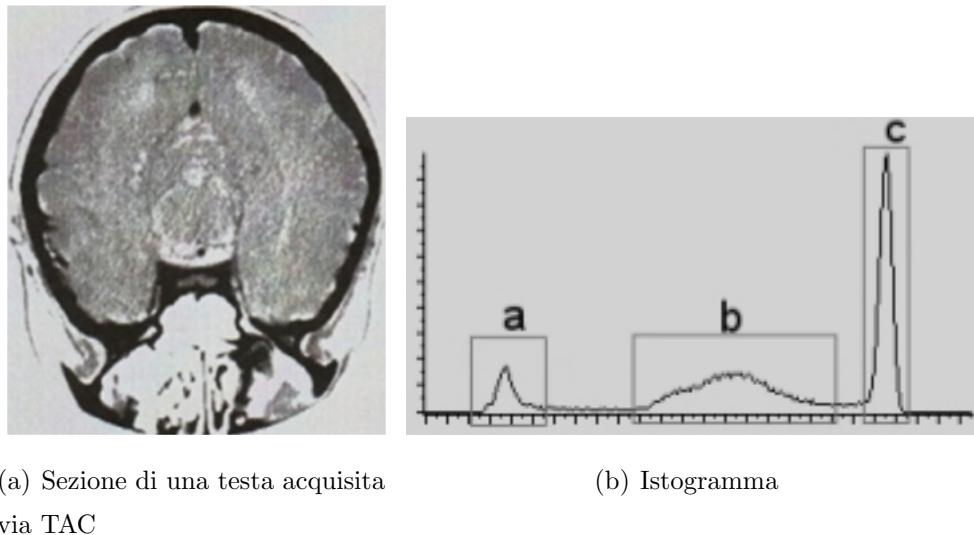


Figura 1.22: Esempio di una immagine composta da tre zone omogenee

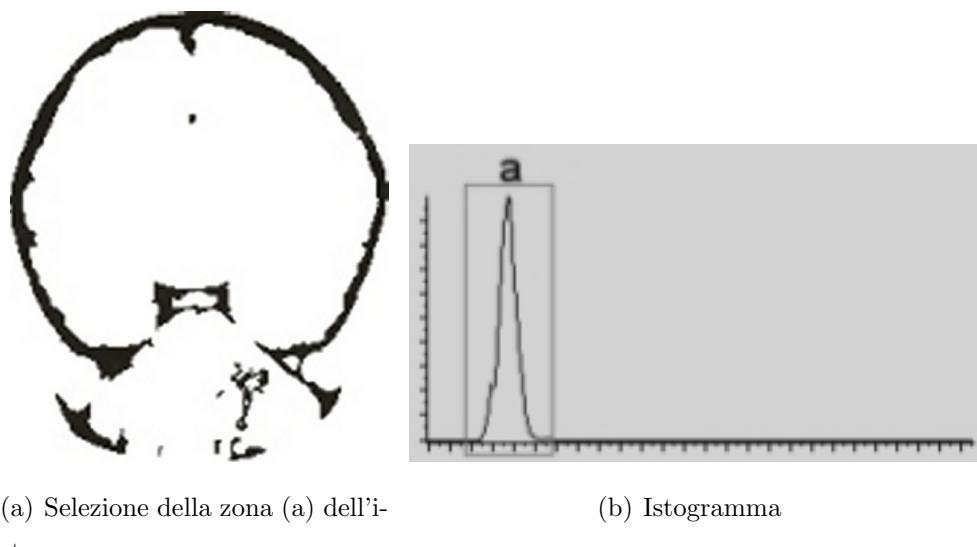


Figura 1.23: Esempio di estrazione di una regione di interesse



Figura 1.24: Immagine binaria

1.4 Ricostruzione e visualizzazione 3D

Una volta che la sequenza delle immagini è stata filtrata, ed attraverso il processo di segmentazione individuata la regione di interesse con il relativo intervallo di soglia $[\mathbf{T}_L, \mathbf{T}_H]$, si procede alla ricostruzione e visualizzazione in tre dimensioni [8].

Vi sono due modi per fare questo:

- attraverso l'estrazione di isosuperfici, Fig 1.25(a).
- modalità diretta Fig 1.25, (b).

Nel primo caso, viene eseguito un passaggio intermedio che permette di estrarre, dall'insieme di dati di tipo volumetrico (voxel), un'informazione di tipo geometrico¹³. Anche in questo caso vi sono due modi di procedere. È possibile estrarre da ciascuna immagine i contorni dell'oggetto di interesse, così da ottenere una pila di contorni che vengono poi uniti con una griglia di polinomi interpolanti. La superficie così ottenuta, viene poi triangolarizzata.

Un approccio diverso, invece, è quello seguito dall'algoritmo del *Marching Cubes* [15, 29] che verrà approfondito in seguito. Una volta ottenuta la superficie, questa viene resa a video attraverso algoritmi di *surface rendering*.

¹³generalmente si tratta di una superficie suddivisa in triangoli.

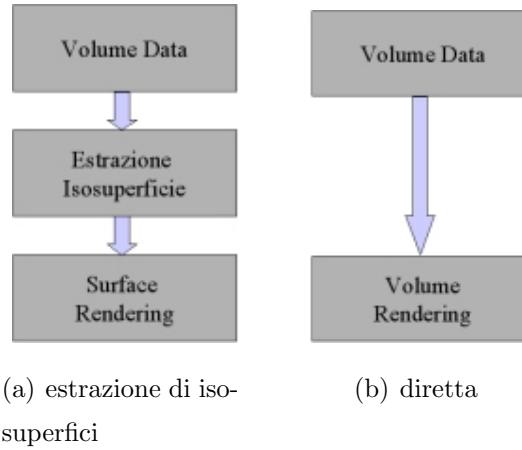


Figura 1.25: Modalità di resa tridimensionale di un set di dati volumetrico

Nella modalità diretta, l’immagine finale viene resa a video direttamente dal set di dati volumetrico. A ciascun voxel viene associato un livello di opacità¹⁴, ed utilizzando le leggi dell’ottica viene proiettato un fascio di raggi (*ray-casting*) [21, 29, 7] che illuminano il volume (Fig 1.26), cosicchè i diversi contributi di attenuazione, offerti dai voxel proiettati sullo stesso pixel del piano immagine, sono miscelati (*blending*) e vanno a generare un pixel dell’immagine finale.

Sono stati sviluppati diversi tipi di modelli ottici, per la propagazione della luce all’interno del volume¹⁵, dove vengono però ignorati alcuni fenomeni come: l’attenuazione che subiscono i raggi che si dirigono verso le sorgenti luminescenti all’interno della gelatina stessa (*shadowing*) e l’ interriflessione (*internal scattering*) tra le varie particelle (*voxel*) in sospensione.

Il rendering volumetrico dello stesso volume di dati utilizzando diverse funzioni di *blending*, è riportato in Fig 1.27, si noti come la scelta della funzione di blending incida sul risultato finale.

Nel prossimo paragrafo verrà affrontato il problema di estrarre l’informa-

¹⁴in funzione del livello di grigio del pixel

¹⁵il volume è modellato fisicamente come una gelatina che mantiene in sospensione delle particelle riflettenti (i voxel)

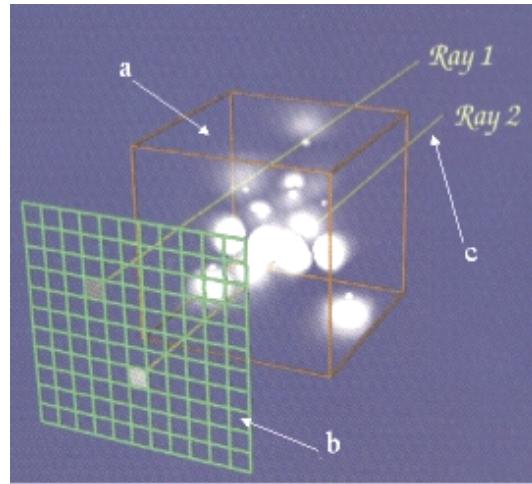


Figura 1.26: a : dati volumetrici; b : piano immagine; c : fascio di raggi



Figura 1.27: Esempio di rendering volumetrico diretto utilizzando diverse funzioni di blending.

zione morfologica di una regione d'interesse, nel caso di un set di dati bidimensionale attraverso l'algoritmo del Marching Squares. Si vedrà poi come una semplice estensione al caso tridimensionale di tale algoritmo porterà all'estrazione di superfici d'interesse da un set di dati volumetrico (Marching Cubes).

1.4.1 Marching squares

Si consideri l'immagine di Fig 1.28.

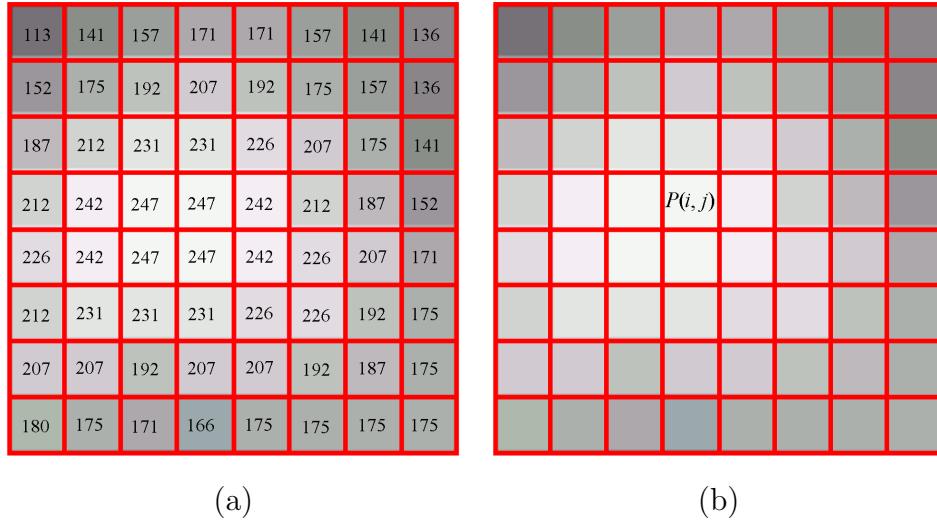


Figura 1.28: Immagine a livelli di grigio. In (a) sono riportati i valori di intensità di grigio. (b) $P(i, j)$ è l'intensità di grigio associato al pixel alla i-esima riga e j-esima colonna

Fissato un intervallo di soglia $[T_L, T_H]$ (determinato nel processo di thresholding), si vuole trovare una linea chiusa (isolinea) che delimita l'insieme dei pixel $P(i, j) \in [T_L, T_H]$, come in Fig 1.29 [29].

L'approccio usato è quello del *divide et impera*. Il problema generale viene diviso in più sottoproblemi ognuno dei quali viene risolto in modo indipendente dagli altri. Unendo poi i singoli risultati si ottiene la soluzione del problema generale cercata.

Si consideri una griglia sovrapposta all'immagine di partenza Fig 1.30 in modo che ad ogni vertice di ciascun quadrato $S(i, j)$ della griglia venga associato un pixel. Quindi al quadrato $S(i, j)$ corrisponderanno i pixel :

$$P(i, j), P(i + 1, j), P(i + 1, j + 1), P(i, j + 1)$$

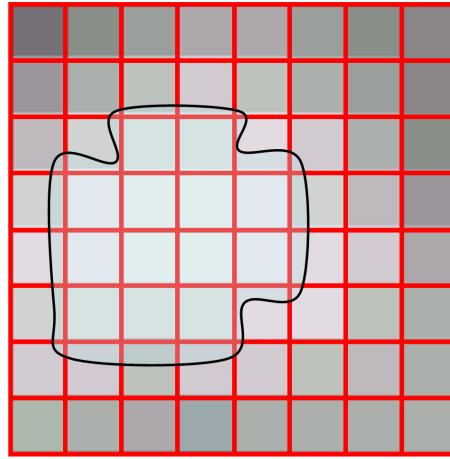


Figura 1.29: Esempio di isolinea definita sull'intervallo $[\mathbf{T}_L, \mathbf{T}_H] = [231, 247]$

Quando un pixel $P(i, j) \in [\mathbf{T}_L, \mathbf{T}_H]$ ¹⁶, ha un pixel vicino $P(i + 1, j)$ ¹⁷ $\notin [\mathbf{T}_L, \mathbf{T}_H]$ ¹⁸, allora una parte della isolinea cercata sicuramente si trova tra i due pixel. Identificando con il pallino verde i pixel $P(i, j) \in [\mathbf{T}_L, \mathbf{T}_H]$ e con il pallino rosso quelli tali che $P(i, j) \notin [\mathbf{T}_L, \mathbf{T}_H]$, per l'immagine data si ottiene la configurazione di Fig 1.31. Ciascun quadrato $S(i, j)$ viene poi classificato in funzione della configurazione assunta, il numero di configurazioni possibili per ciascun quadrato sono 2^4 (Fig 1.32), quattro vertici, ognuno con due possibili stati e viene quindi calcolato il segmento di isolinea (segmento blu in Fig.1.33) attraverso una interpolazione sui livelli di grigio assunti dai pixel.

Sia X la posizione spaziale del vertice del quadrato $S(i, j)$ a cui è associato il pixel $P(i, j)$, allora gli estremi X_1 e X_2 , che definiscono il segmento di isolinea cercato, sono calcolati come:

$$X_1 = X + \left(\frac{T - P(i, j)}{P(i + 1, j) - P(i, j)} \right) \quad (1.4)$$

¹⁶il pixel è dentro la regione di interesse

¹⁷oppure $P(i,j+1)$, $P(i+1,j+1)$

¹⁸il pixel è fuori la regione di interesse

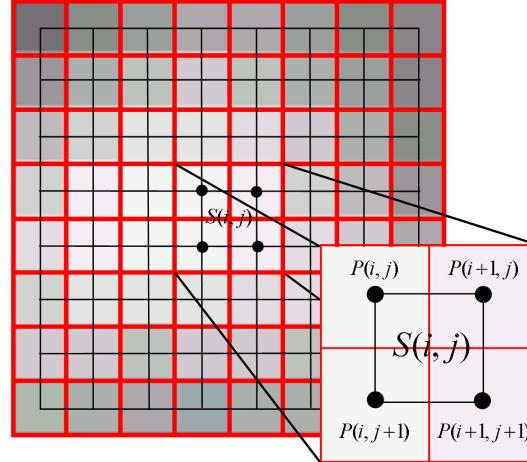
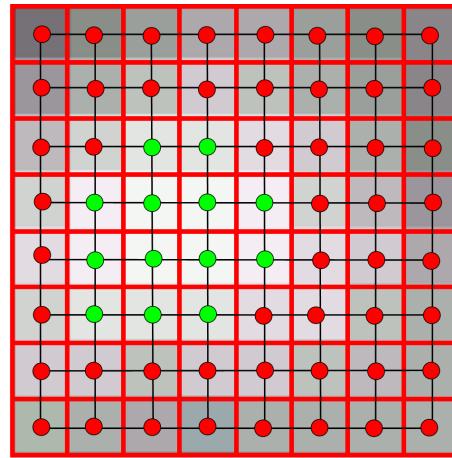


Figura 1.30: Posizionamento della griglia sull'immagine di Fig 1.28

$$X_2 = X + \left(\frac{T - P(i, j)}{P(i, j + 1) - P(i, j)} \right) \quad (1.5)$$

dove $T = T_L$ se $P(i, j) \leq T_L$, oppure $T = T_H$ se $P(i, j) \geq T_H$. Infine unendo tutti i segmenti ottenuti si ha l'isolinea cercata Fig 1.34.

Figura 1.31: Classificazione dei vertici di ciascun quadrato $S(i, j)$ ottenuta usando come intervallo di soglia $[T_L, T_H] = [231, 247]$

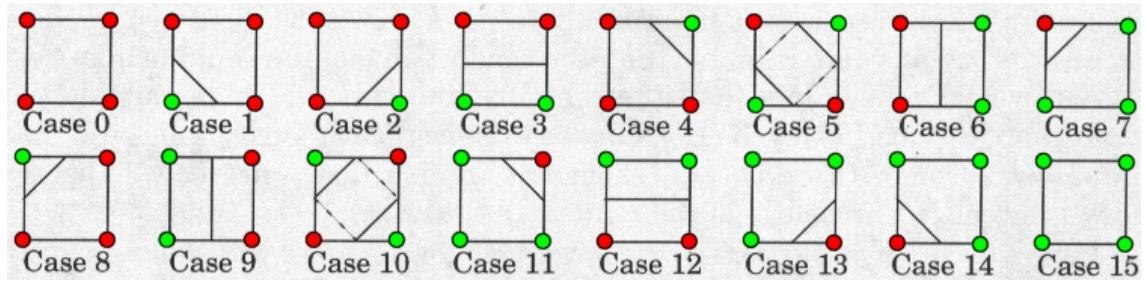


Figura 1.32: Configurazioni possibili per ciascun quadrato

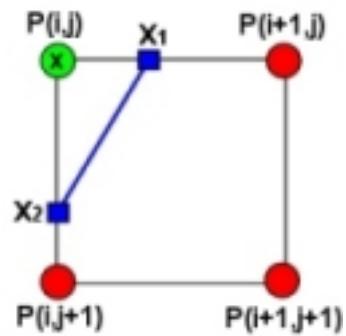
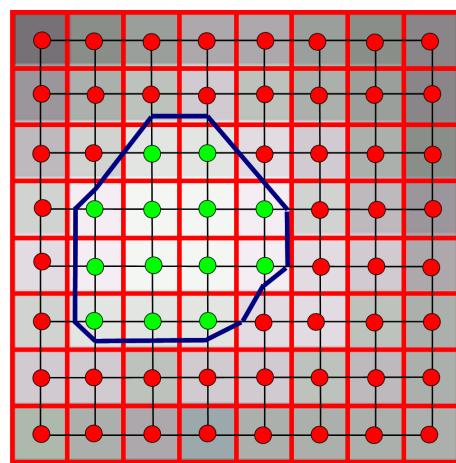
Figura 1.33: Calcolo del segmento di isolinea relativo al quadrato $S(i,j)$ 

Figura 1.34: Isolinea cercata

1.4.2 Marching cubes

L'algoritmo appena descritto può essere facilmente esteso al caso tridimensionale, cioè quando si ha una sequenza di immagini. In tal caso vengono utilizzati cubi¹⁹, anzichè quadrati.

Data una sequenza di immagini e fissato un intervallo di soglia $[\mathbf{T}_L, \mathbf{T}_H]$ si vuole trovare una **superficie** (isosuperficie) che delimita l'insieme dei *voxel*²⁰ tali che $V(i, j) \in [\mathbf{T}_L, \mathbf{T}_H]$.

Vengono elaborate due immagini alla volta : date le immagini k e $k + 1$ viene sovrapposta una griglia di cubi tale da far corrispondere ai vertici di ciascuno $C(i, j, k)$ quattro pixel dell'immagine k e quattro della $k + 1$ (Fig 1.35).

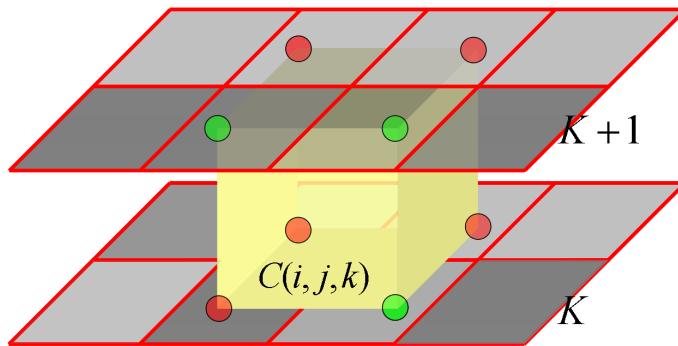


Figura 1.35: Classificazione dei voxel

In questo caso il numero delle possibili configurazioni per ciascun cubo sono 2^8 , che però possono essere ridotte a 15 eliminando le configurazioni equivalenti, perchè ottenute come rotazione o inversione di configurazioni base.

Si vede come il cubo $C(i,j,k)$, riportato in Fig 1.36 (a) corrisponda alla configurazione n°5 in Fig 1.37 ed il contributo fornito alla isosuperficie cercata

¹⁹da cui *marching cubes*

²⁰è da ricordare che quando si ha una sequenza di immagini, che rappresentano sezioni di un volume, a ciascun pixel $P(i,j)$ rappresenta un elemento di volume $V(i,j)$ delle dimensioni definite nell'header del file DICOM

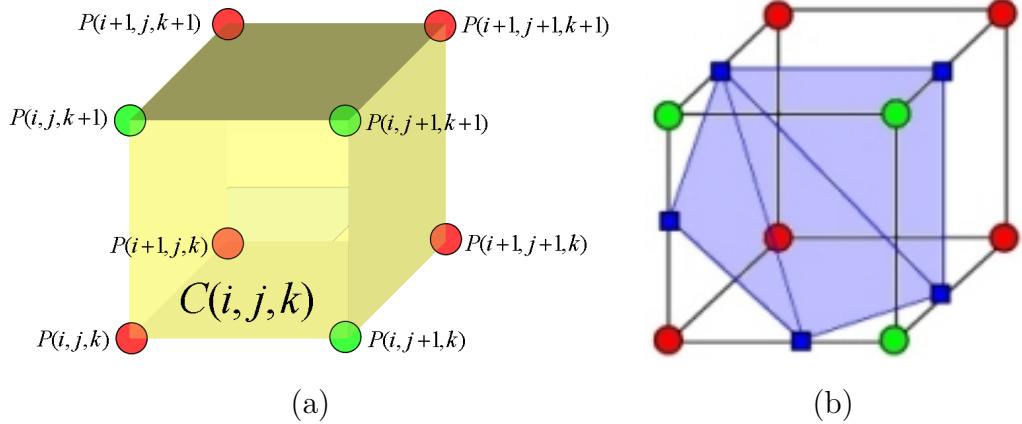


Figura 1.36: (a) classificazione dei vertici del cubo $C(i, j, k)$. (b) contributo di isosuperficie generata da $C(i,j,k)$.

è l'insieme di triangoli di Fig 1.36 (b). Iterando il procedimento per tutte le immagini ed unendo i triangoli ottenuti, si arriva alla superficie triangolarizzata cercata. Spesso, però, il numero di triangoli generati in questo modo è molto alto e per questo che si rende sempre necessaria un' operazione in *post-processing* di decimazione, il cui scopo è quello di ridurre il numero di triangoli componenti la isosuperficie. Tale operazione si ripercuote, ovviamente, sulla qualità della superficie, ma in questo modo viene sensibilmente ridotto il tempo di rendering, che è proporzionale al numero di triangoli della superficie. L'operazione di decimazione è molto importante per il rendering visio-aptico, dove non possono essere usati modelli tridimensionali particolarmente complessi. Alla decimazione segue un'operazione di *surface smoothing* [4, 19] capace di rimuovere piccoli artefatti ed irregolarità dovute al processo ricostruttivo e di decimazione. Un esempio di isosuperficie è riportato in Fig 1.38.

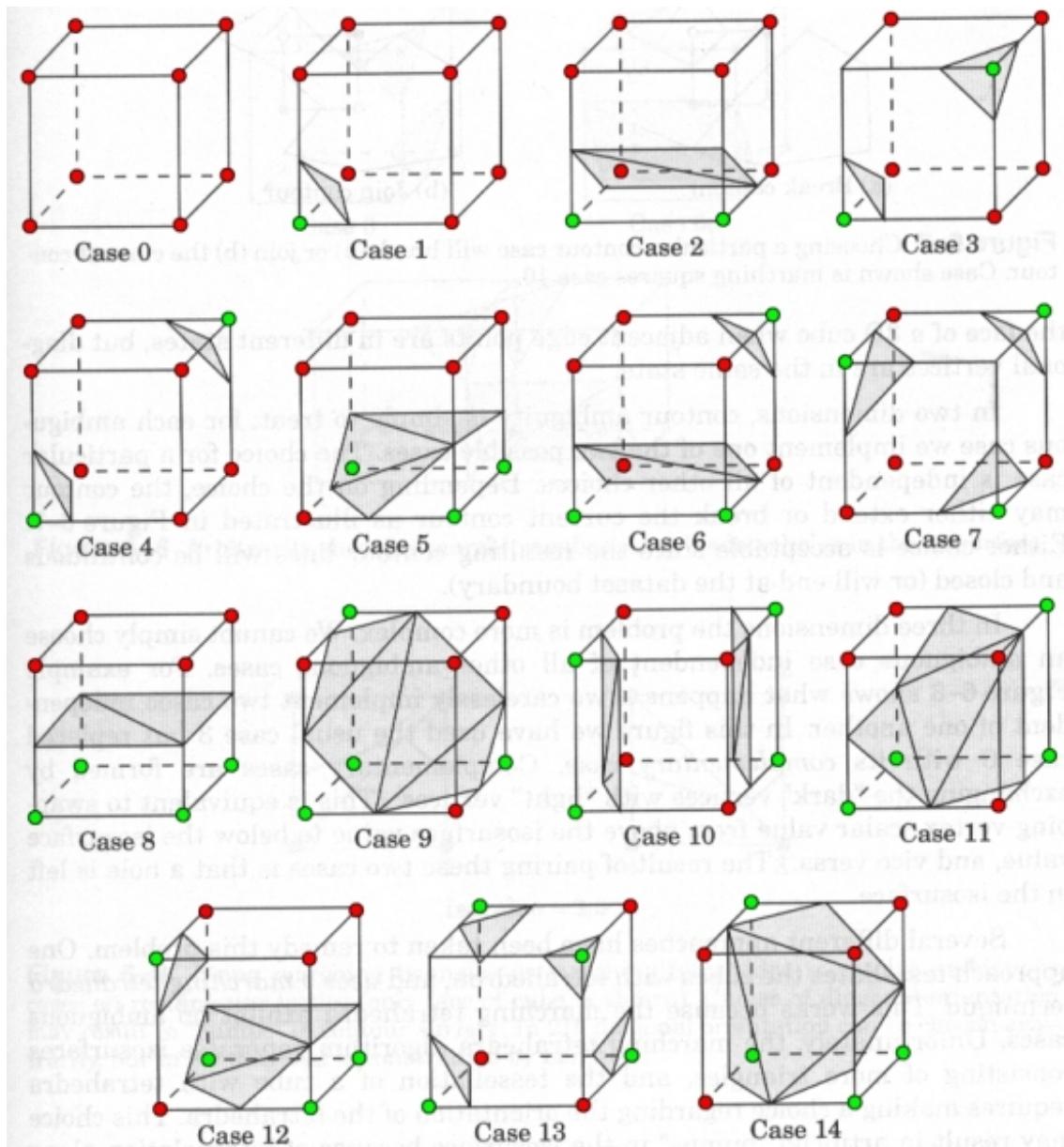


Figura 1.37: Tabella di classificazione dei cubi

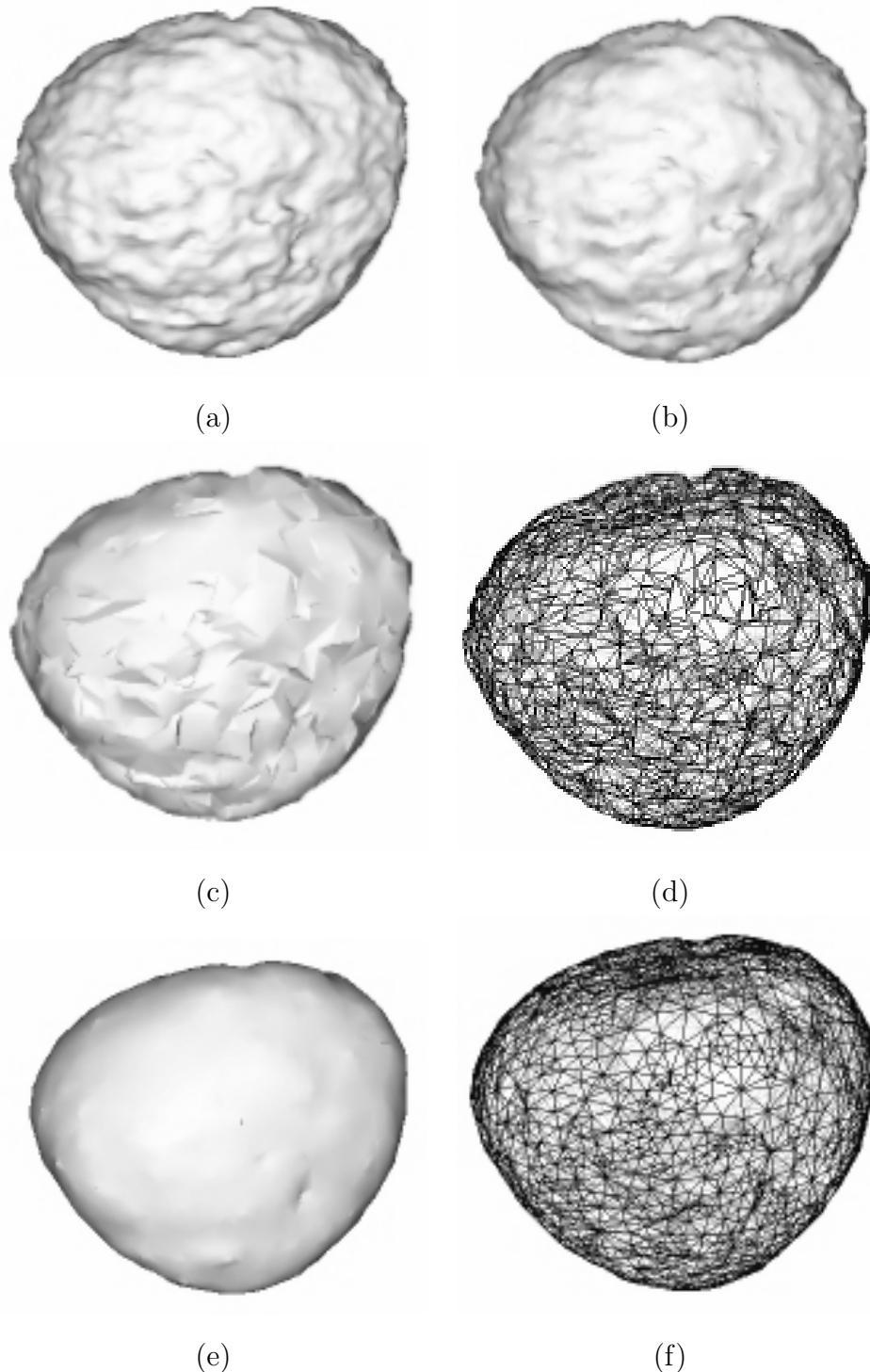


Figura 1.38: Esempio di isosuperficie estratta da un set di dati volumetrico. Si può osservare come incide sulla superficie il processo di decimazione e quello di smoothing. (a) 66872 triangoli, (b) decimazione 50% : 31874 triangoli, (c) decimazione 90% : 5610 triangoli, (e) caso (c) con smoothing 50%

Capitolo 2

Interfacce aptiche

Le interfacce aptiche costituiscono un importante passo avanti nel migliorare l’interazione uomo macchina. La possibilità di provare una sensazione tattile di un qualcosa che nella realtà non esiste e che prima era solo possibile percepire attraverso il senso visivo, aumenta il flusso informativo macchina-uomo e quindi le percezioni di quest’ultimo durante l’esplorazione di una realtà virtuale. In questo capitolo verranno affrontante le diverse problematiche che si presentano nella progettazione di un sistema *visivo-tattile*. In particolare saranno discusse le problematiche relative alla *collision detection*, *force rendering* e *proxy*.

2.1 Psicofisica

Per progettare un sistema che fa uso dei sensi è innanzitutto necessario capire come questi funzionino [20]. In questo ci aiutano gli studi di psicofisica e neurofisiologia che studiano come un soggetto, attraverso processi neurali e percettivi, interagisce con ambienti reali. Con tali studi si è stimato, per esempio, che l’occhio produce un flusso informativo pari a 10^6bit/sec , mentre l’orecchio di 10^4bit/sec . Di nostro interesse è il senso tattile, inteso non come la sensazione prodotta dallo sfioramento di una superficie, ma come quella cinestatica

prodotta come reazione dal sistema muscolare¹. Alla sensazione cinestatica è associata una larghezza di banda del flusso informativo pari a $20Hz$ - $30Hz$. Questo ci fa capire come un qualunque dispositivo in grado di eccitare il senso cinestatico, tramite feedback di forza sull'utente, debba essere aggiornato con una frequenza di almeno $30Hz$ per rendere realistica la sensazione tattile.

2.2 Note storiche

L'utilizzo delle interfacce aptiche si è reso necessario per la prima volta dopo la seconda guerra mondiale, nell'industria nucleare. Data la dannosità dei materiali radioattivi si rendeva necessario progettare un sistema che ne permetesse la manipolazione in via remota. Così nel 1947 fu costituito all'*Argonne National Laboratory* l'*RCG*² che mise a punto un primo sistema aptico costituito da un insieme di bracci collegati in modo puramente meccanico. I *link* furono sostituiti poi da sistemi controllati elettricamente, per giungere poi, ad un vero e proprio sistema robotico. Questo segnò la nascita della manipolazione *telerobotica*. Negli anni si è poi assistito alla nascita, all'interno dei laboratori di ricerca, di diversi tipi di interfacce aptiche con diversi gradi di libertà e funzionalità. Finchè nel 1993 fecero la comparsa sul mercato le prime interfacce aptiche commerciali, Touch Master e SAFiRE Master prodotte dalla *EXOS Inc*, nate per l'interazione con ambienti virtuali. Seguì nel 1995 il PHANToM della *SensAble Technologies Inc*.

¹si provi ad esercitare con un dito una forza su una superficie e a sentirne la forza di reazione, questa è il senso tattile cinestatico

²Remote Control Group

2.3 Il PHANToM

L’interfaccia aptica utilizzata in questa tesi è il **PHANToM** [35] (Personal HAptic iNTERface Mechanism) della SensAble Technologies Inc. Il PHANToM è un manipolatore robotico con tre gradi $\theta_0, \theta_1, \theta_2$ di libertà per il force feedback. Vi sono anche altri tre gradi di libertà di rotazione (Roll,Pitch,Yaw) forniti dall’end effector, questi però non sono attuati e quindi non dispongono di force feedback. In Tab 2.1 sono riportate le caratteristiche tecniche del PHANToM. Per sentire una forza $\mathbf{F} = [f_x f_y f_z]^T$ attraverso una interfaccia ap-

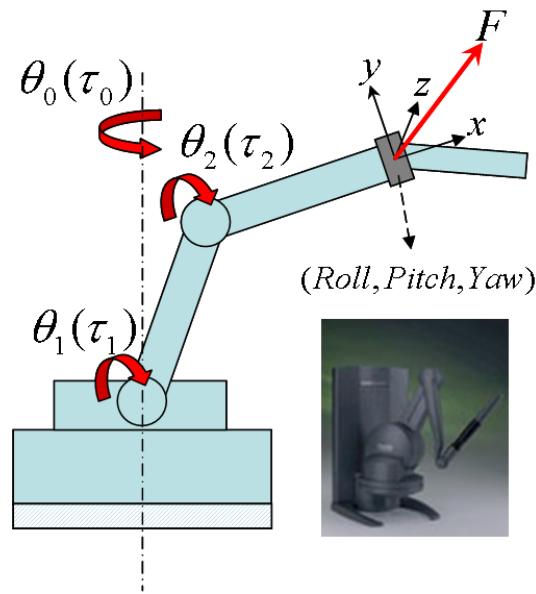


Figura 2.1: PHANToM

tica, questa deve prima essere trasformata in una serie equivalente di coppie $\mathbf{\Gamma} = [\tau_0 \tau_1 \tau_2]^T$ da applicare ai motori (giunti rotazionali). La coppia corrispondente alla forza \mathbf{F} , viene calcolata tramite la matrice jacobiana trasposta \mathbf{J}^T , che mappa le forze espresse nel sistema cartesiano dell’end-effector, nelle equivalenti di torsione sui giunti rotazionali. La matrice jacobiana è legata alla

cinematica dell’interfaccia aptica.

$$\boldsymbol{\Gamma} = \boldsymbol{J}^T \mathbf{F} \quad (2.1)$$

Nominal position resolution	0.02 mm
Workspace	16 x 13 x 13 cm
Backdrive friction	0.06 N
Maximum exertable force	6.4 N
Continuous exertable force (24 hrs)	1.7 N
Stiffness	3.16 N/mm
Inertia (apparent mass at tip)	<75 g
Footprint	18 x 16 cm
Force feedback	3 degrees of freedom (x, y, z)
Position sensing	6 degrees of freedom (x, y, z, roll, pitch, yaw)
Input voltage	90-260 VAC
Input frequency	47-63 Hz
Input current	2A @ 115 VAC 1A @ 230 VAC
System Requirements	Windows 2000, or Red Hat Linux 7.2

Tabella 2.1: Specifiche PHANToM. Copyright 1995, SensAble Technologies, Inc. All rights reserved.

2.4 Collision detection

Generalmente un ambiente virtuale può essere statico o dinamico. Nel primo caso tutti gli oggetti occupano una posizione fissa, non vi sono pertanto variazioni della scena nel tempo. Nel secondo caso invece vi è almeno un oggetto dinamico, cioè un'entità che modifica il suo stato nel tempo e come tale influenza anche sullo stato dell'intera scena. Questa deve quindi essere aggiornata periodicamente in funzione delle interazioni che si presentano tra le diverse entità. Un punto fondamentale e di primaria importanza per calcolare le interazioni, è quello di verificare quando questi oggetti entrano in contatto. Ciò equivale a fare un test di collision detection [9, 20, 38, 10](Fig 2.2).

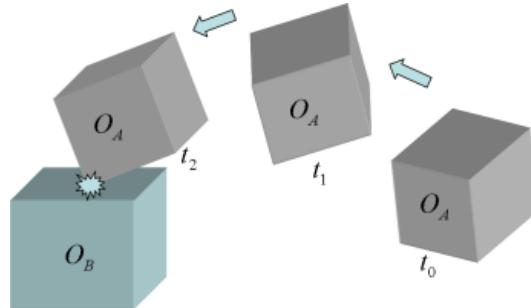


Figura 2.2: Scena virtuale composta da due oggetti, uno statico O_B ed uno dinamico O_A che va a collidere al tempo t_2 con O_B

In questa tesi si considera l'ambiente virtuale costituito da un oggetto statico che rappresenta il modello tridimensionale che si vuole toccare e da un oggetto dinamico rappresentativo della posizione dell'interfaccia aptica. Il problema è quindi quello di sapere ad istanti di tempo regolari, se l'interfaccia è entrata o no, in collisione con gli oggetti statici dell'ambiente virtuale. Un approccio semplicistico e poco efficiente al problema, è quello di testare, ad ogni ciclo di controllo, tutti i triangoli che compongono la superficie dell'oggetto (Fig 2.3). Considerando però che il numero di triangoli per un oggetto complesso è dell'ordine delle decina di migliaia, il tempo necessario per tale

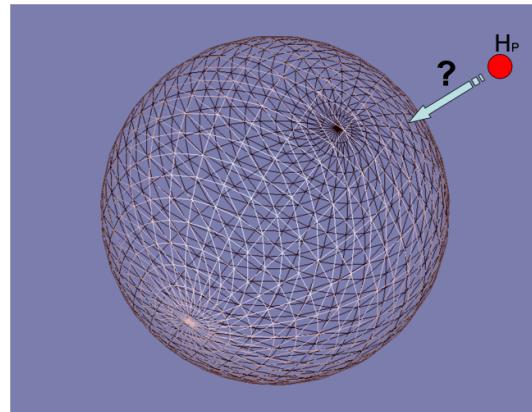


Figura 2.3: Collision detection. H_P : Posizione dell’interfaccia aptica

operazione sarebbe troppo lungo. Infatti è necessario dire che all’operazione di collision detection segue quella della generazione della forza (processo di force rendering) e tutto questo va eseguito, per il PHANToM, in un tempo minore del millisecondo (questo per garantire la stabilità dell’interfaccia) (Fig 2.4). In

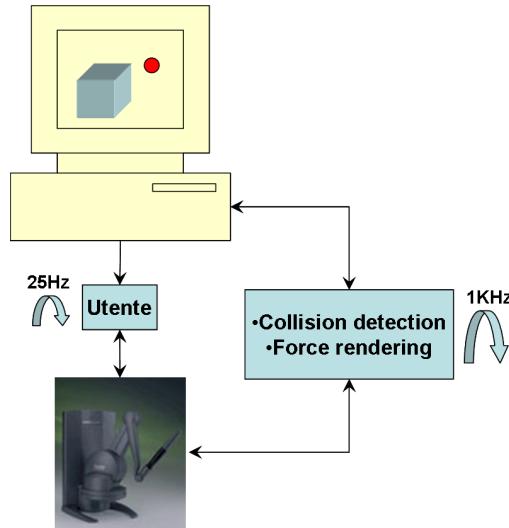


Figura 2.4: Processo aptico

questa tesi si è cercata una soluzione che risolvesse questi problemi. La tecnica

di collision detection implementata fa uso dell’OBB-Tree (Oriented Bounding Box Tree).

2.4.1 OBB-Tree

L’idea è quella di eseguire una scomposizione spaziale della superficie complessa (cioè composta da un elevato numero di triangoli) utilizzando delle primitive geometriche, come, cubi, sfere, coni, etc, che vengono poi organizzate in una struttura gerarchica (albero binario di ricerca) Fig 2.5.

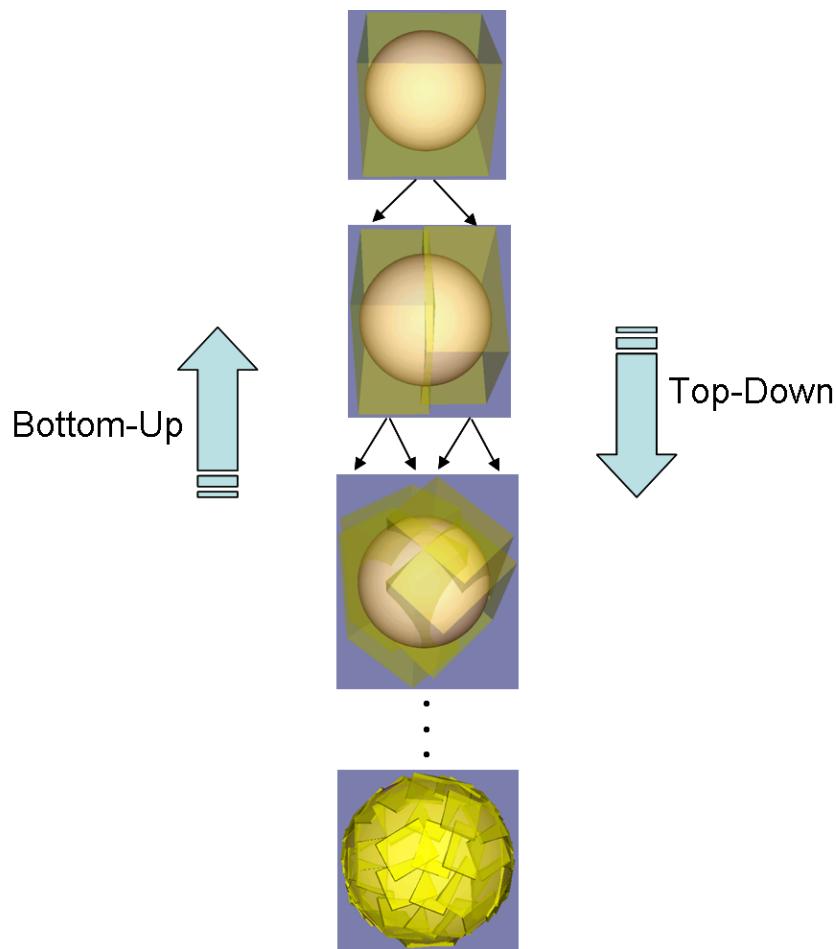


Figura 2.5: OBBTree

Per fare questo vi sono due diversi approcci: bottom-up e top-down.

Il primo, inizialmente prevede di associare a ciascun triangolo della superficie un OBB, per poi eseguire su questi delle operazioni di merging, in modo da ottenere degli OBB più grandi. Il processo termina quando si ottiene un OBB che ingloba tutta la superficie, il quale andrà poi a costituire la radice dell'albero di ricerca. Mentre gli OBB inizialmente calcolati, ne costituiranno le foglie.

Nel secondo caso invece è previsto che inizialmente venga calcolato l'OBB contenente l'intera superficie, cioè la radice dell'albero, che sarà poi ricorsivamente suddiviso in due parti (operazione di splitting), finché ad ogni triangolo non sarà associato un OBB. In questo caso si parte dalla radice per arrivare alle foglie. Generalmente però non si arriva ad un tale livello di splitting, ma viene fissato un livello di soglia sul numero massimo di triangoli che un OBB foglia dell'albero deve contenere. L'approccio seguito in questa tesi è quello top-down.

L'algoritmo di calcolo degli OBB fa uso dei momenti statistici del primo e secondo ordine quindi media (μ) e matrice di covarianza (C).

Sia $S = \{\mathbf{V}_i, 0 \leq i < n\}$ l'insieme dei vertici \mathbf{V}_i dei triangoli costituenti la superficie. Allora si ha:

$$\mu = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{V}_i \quad (2.2)$$

$$C = \frac{1}{n} \sum_{i=0}^{n-1} (\mathbf{V}_i - \mu)^T (\mathbf{V}_i - \mu) \quad (2.3)$$

Gli autovettori \mathbf{a}_i , \mathbf{a}_j , \mathbf{a}_k , di una matrice simmetrica, come C , sono mutualmente ortogonali. Pertanto, una volta normalizzati, vengono usati come base (Oriented) per l'OBB (Fig 2.6), che includerà tutti i vertici utilizzati per calcolare la matrice C . Invece per dimensionare (Bound) l'OBB lungo le tre direzioni viene preso il massimo e il minimo delle proiezioni dei vertici, usati per il calcolo della matrice di covarianza (C), sui tre assi \mathbf{a}_i , \mathbf{a}_j , \mathbf{a}_k . Il centro

dell'OBB è poi posizionato in μ . Una volta calcolato un OBB si controlla

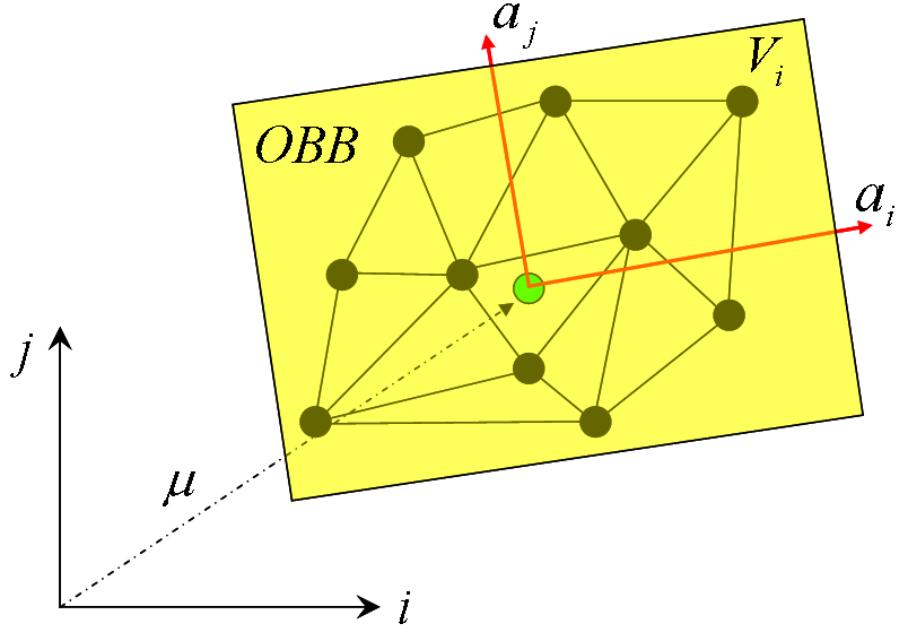


Figura 2.6: Calcolo di un OBB

se il numero di vertici contenuti è minore delle soglia fissata. Se è così l'OBB non è più ulteriormente suddiviso. Altrimenti ne viene eseguito lo splitting, cosicchè la superficie contenuta viene suddivisa in due sottosuperficie e di queste vengono calcolati i rispettivi OBB. La procedura di splitting prevede di dividere l'OBB con un piano di sezione ortogonale al lato più lungo. L'insieme dei vertici viene poi partizionato in base alla posizione che hanno rispetto alle normali del piano.

Il problema è però quello di determinare la posizione del piano di sezione. Vi sono diverse euristiche per fare questo, due delle quali sono:

1. posizionare il piano di sezione a metà del lato più lungo. In questo modo però l'albero non viene bilanciato. Questo è dovuto alle diverse aree assunte dai triangoli, che comportano una non uniforme distribuzione dei vertici.

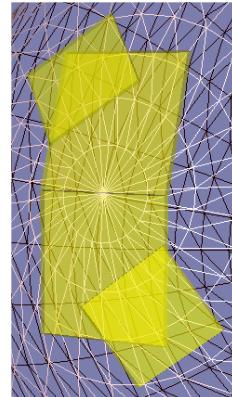


Figura 2.7: Particolare di tre OBB che tassellano una superficie sferica

2. posizionare il piano di sezione nel punto corrispondente al valore medio della proiezione dei vertici, lungo l'asse più lungo. Questo garantisce un bilanciamento dell'albero, quindi una profondità minore e tempi di ricerca ottimizzati.

2.4.2 Usare l'OBBTree per la collision detection

Vediamo ora come utilizzare l'OBB-Tree per implementare il processo di collision detection. Nota istante per istante la posizione dell'interfaccia aptica (H_P), si vuole sapere quando questa penetra l'oggetto da toccare. L'idea è quella di verificare se il segmento, congiungente la posizione attuale e quella precedente (H_{LP}), interseca o meno un OBB foglia dell'albero (Fig 2.8). Per questo si controlla prima se il segmento interseca l'OBB radice, se non c'è intersezione allora non si è avuta alcuna collisione. Altrimenti, se c'è intersezione, si prosegue la ricerca nell'albero e si verifica quale dei due OBB figli, destro o sinistro, è intersecato. Se nessuno dei due lo è si conclude il processo di ricerca ed ancora una volta non si è avuta collisione. Se invece uno degli OBB figli, per esempio il sinistro, è intersecato, allora si prosegue nel sottoalbero sinistro, eliminando così tutto l'insieme di triangoli che costituisce la parte di oggetto

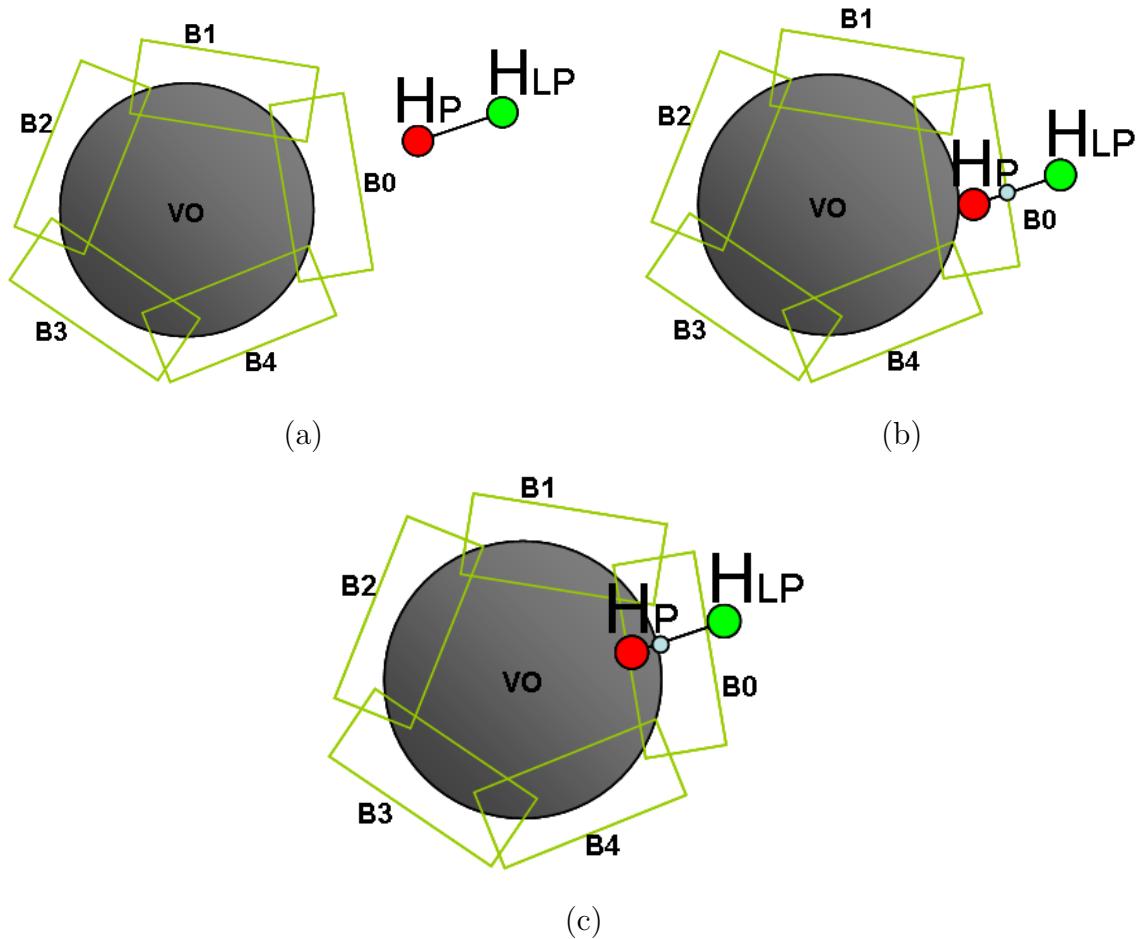


Figura 2.8: Test di collisione detection : (a) Nessuna collisione. (b) Collisione solo con l'OBBTree. (c) Collisione con l'oggetto VO (Virtual Object).

contenuta nel sottoalbero di destra. Se il processo di ricerca termina su un nodo intermedio, si può dire sicuramente che non si è avuta collisione, se invece si conclude su una foglia, per esempio B_0 , quindi non vi sono ulteriori OBB da controllare, si vanno a testare tutti e soli i triangoli contenuti in B_0 , che generalmente sono dell'ordine delle decina, il test è pertanto immediato ed è quindi possibile sapere se c'è stata intersezione, qual'è il triangolo intersecato ed il punto di intersezione.

2.5 Proxy

Quando si verifica una collisione, il punto H_P si trova, nella parte interna di VO (Fig 2.8 (c)). A questo punto è necessario calcolare, a partire da H_P , un punto situato sulla superficie esterna a VO che consenta di :

- calcolare la forza da attuare sull'operatore (processo di force rendering), che è proporzionale alla penetrazione nell'oggetto.
- visualizzare un punto che si muove sulla superficie esterna di VO, anziché la reale posizione dell'interfaccia aptica che è interna al VO, per non dare all'utente la sensazione di aver penetrato l'oggetto.

Il punto in questione è noto con il nome di proxy. In letteratura l'algoritmo per il calcolo del proxy più comune è quello del god-object [27, 26]. In questa tesi si è voluto studiare un algoritmo, per il calcolo del proxy, che avesse un basso carico computazionale. L'idea di base è stata quella di utilizzare le informazioni fornite dal processo di collision detection. Pertanto, la prima volta che si verifica una collisione il proxy P_0 , riportato in Fig 2.9, viene semplicemente calcolato come il punto di intersezione tra il segmento $\overline{H_{LP}H_P}$ ed il VO. Supponiamo adesso, che l'utente muova l'interfaccia aptica dalla posizione H_P alla nuova posizione H_{NP} , il nuovo proxy P_1 sarà calcolato come il punto di intersezione

tra il segmento $\overline{NH_{NP}}$ ed il VO, dove :

$$\mathbf{N} = \mathbf{P}_0 + \mathbf{T} \quad (2.4)$$

\mathbf{N} è il punto ottenuto traslando il proxy \mathbf{P}_0 , calcolato al passo precedente, lungo la direzione della componente tangente (\mathbf{T}) la superficie in \mathbf{P}_0 del vettore spostamento, \mathbf{L} , del PHANTOM. Il vettore \mathbf{T} viene calcolato come differenza tra il vettore spostamento \mathbf{L} e la componente di quest'ultimo lungo la normale $\hat{\mathbf{n}}$ alla superficie in \mathbf{P}_0 , \mathbf{L}_n .

$$\mathbf{T} = \mathbf{L} - \mathbf{L}_n \quad (2.5)$$

$$\mathbf{L}_n = \langle \mathbf{L}, \hat{\mathbf{n}} \rangle \hat{\mathbf{n}} \quad (2.6)$$

$$\mathbf{L} = \mathbf{H}_{NP} - \mathbf{H}_P \quad (2.7)$$

E' da notare come il processo di intersezione necessario ad aggiornare il proxy non introduca un overhead nel processo di gestione del PHANTOM (1KHz), infatti questo è lo stesso test di intersezione usato nel controllo di collision detection. Quindi il risultato ottenuto è stato quello che attraverso una scelta opportuna del segmento attraverso il quale si esegue il test di collision detection , utilizzando l'OBB-Tree, è possibile testare le collisioni e contestualmente aggiornare il proxy.

2.6 Force rendering

Una volta verificata la collisione con l'oggetto (VO) e calcolato il nuovo proxy (P1, Fig 2.9) siamo nella condizione di poter calcolare la forza da trasmettere all'operatore per il rendering aptico [26]. C'è quindi bisogno di definire un modello fisico del VO. Le due tipologie possibili sono:

- Oggetti cedevoli
- Oggetti rigidi

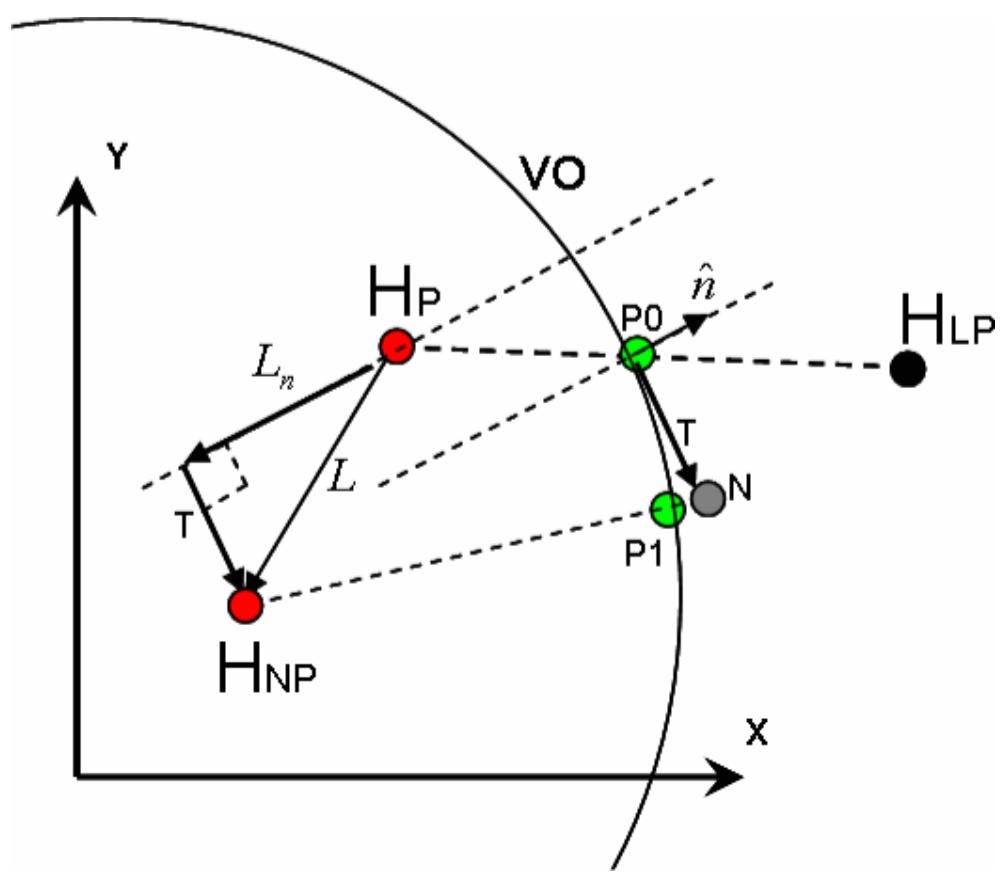


Figura 2.9: Calcolo del proxy

Per i primi sono disponibili in letteratura diverse tecniche[25, 40, 12], il problema fondamentale è il pesante carico computazionale richiesto per elaborare le deformazioni e quindi il feedback di forza. Per questo, tipicamente, l'implementazione di un sistema visio-aptico con oggetti cedevoli complessi richiede due workstation. Una che gestisca la grafica, e l'altra che si occupi di elaborare il modello dinamico dell'oggetto cedevole (Fig 2.10).

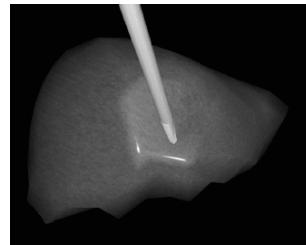


Figura 2.10: Esempio di oggetto cedevole

Il modello usato in questa tesi è di tipo rigido. La sensazione visio-aptica corrisponde a quella di toccare un oggetto rigido.

La forza generata è composta da due componenti, una normale alla superficie nel proxy, \mathbf{F}_n , che fornisce la sensazione cinestatica (la durezza) del tatto e l'altra tangente la superficie nel proxy di viscosità, \mathbf{F}_v . La forza totale \mathbf{F} attuata sul PHANToM, e quindi sull'operatore, è:

$$\mathbf{F} = \mathbf{F}_n + \mathbf{F}_v \quad (2.8)$$

A sua volta la componente normale della forza \mathbf{F}_n è costituita da due componenti. Una elastica, caratterizzata dalla costante K , con la quale viene modellata la “durezza” dell'oggetto e l'altra di smorzamento viscoso. caratterizzata dalla costante di viscosità D . Quest'ultima componente è necessaria per smorzare le oscillazioni prodotte dalla sola componente elastica, in quanto non sarebbe realistico che un oggetto rigido produca una forza di reazione oscillatoria.

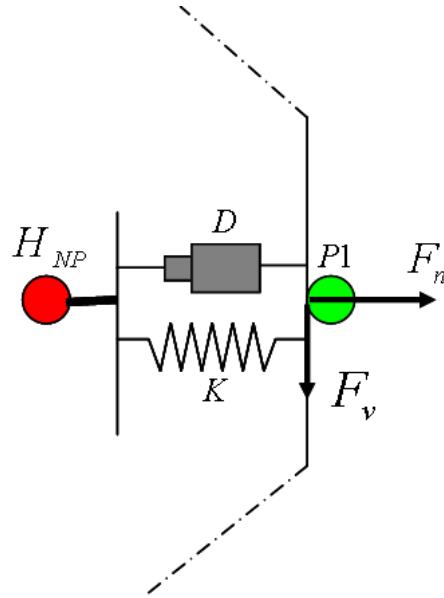


Figura 2.11: Modello fisico del sistema usato per il processo di force rendering.

Si può osservare come la forza elastica sia proporzionale alla penetrazione nell'oggetto Δ_n eseguita lungo la direzione normale $\hat{\mathbf{n}}$, che poi non è altro che la norma del vettore \mathbf{L}_n . Mentre la forza di smorzamento sia proporzionale a \mathbf{v}_n che è la componente lungo $\hat{\mathbf{n}}$ del vettore velocità \mathbf{v} del PHANTOM.

Infine \mathbf{F}_v produce un effetto di viscosità sulla superficie e risulta essere proporzionale alla componente tangente, \mathbf{v}_t , la superficie nel proxy della velocità \mathbf{v} . In Fig 2.9 è riportato il modello fisico usato per il processo di force rendering.

$$\mathbf{F}_n = K\Delta_n - D\mathbf{v}_n \quad (2.9)$$

$$\Delta_n = \|\mathbf{L}_n\| \quad (2.10)$$

$$\mathbf{v}_n = \langle \mathbf{v}, \hat{\mathbf{n}} \rangle \hat{\mathbf{n}} \quad (2.11)$$

$$\mathbf{F}_v = -D\mathbf{v}_t \quad (2.12)$$

$$\mathbf{v}_t = \mathbf{v} - \mathbf{v}_n \quad (2.13)$$

Capitolo 3

Utilizzo delle interfacce aptiche in medicina

Fino a qualche anno fa lo sviluppo e la commercializzazione di sistemi basati su tecnologia aptica erano rivolti soprattutto al settore ludico, in alcuni semplici sistemi di interfacciamento dotati di un feedback di forza che rendesse più reale l'immersione sensoriale nella realtà virtuale.

Negli ultimi anni invece la possibilità di disporre di una notevole potenza di calcolo a prezzi relativamente bassi e lo sviluppo digitale dei sistemi medici hanno portato alla diffusione, anche commerciale, di nuovi sistemi aptici pensati per il campo medico.

L'utilizzo delle tecnologie aptiche in ambito medico è finalizzato al potenziamento dei sistemi di indagine ed allo sviluppo di nuove tecnologie di tipo chirurgico. Questo sta portando, ad una rivoluzione dei sistemi diagnostici e di pianificazione d'intervento.

Un esempio concreto in ambito chirurgico è dato dal sistema della Intuitive Surgical [32] (Fig 3.1), un sistema robotizzato che permette ad un operatore, anche in modo remoto, di intervenire su un paziente con modalità molto meno invasive e più precise. L'operatore è posto in una cabina di monitoraggio ed è



Figura 3.1: Sistema chirurgico robotizzato della Intuitive Surgical.

dotato di un visore, che gli fornisce una visione stereo della zona di intervento e di ditali robotizzati (Fig 3.2 (a)) forniti di feedback di forza che gli consentono di “sentire” la cedevolezza dei tessuti, attraverso cui comanda i microbisturi (Fig 3.2 (b)) che vanno ad operare in via laparoscopica sul paziente.

Tali tecnologie sono già state utilizzate, come a Strasburgo presso l’European Institute of Telesurgery (Etis), dove la chirurgia del futuro è già realtà. E’ stata eseguita, infatti, la prima operazione transoceanica della storia della medicina, i chirurghi di New York hanno rimosso, grazie a sofisticati robot guidati da computer, la colecisti di una signora che si trovava a Strasburgo.

Le potenzialità di una automazione così spinta sono enormi, come ha sottolineato la rivista “Nature” [1] che ha pubblicato la relazione dell’intervento, e riguardano soprattutto la possibilità di limitare l’errore umano, di operare laddove non ci sono strutture adeguate e di ampliare la diffusione delle tecniche più innovative, perché qualunque chirurgo potrà presto apprendere, rimanendo comodamente seduto davanti al proprio computer.

Altre applicazioni interessanti sono da annoverare nel campo della simulazione chirurgica. Sul piano didattico strumenti di questo tipo stanno rivoluzionando l’apprendimento dei giovani medici. Infatti ad oggi la chirurgia è una tecnica manuale imparata sui libri, mentre strumenti di questo tipo possono

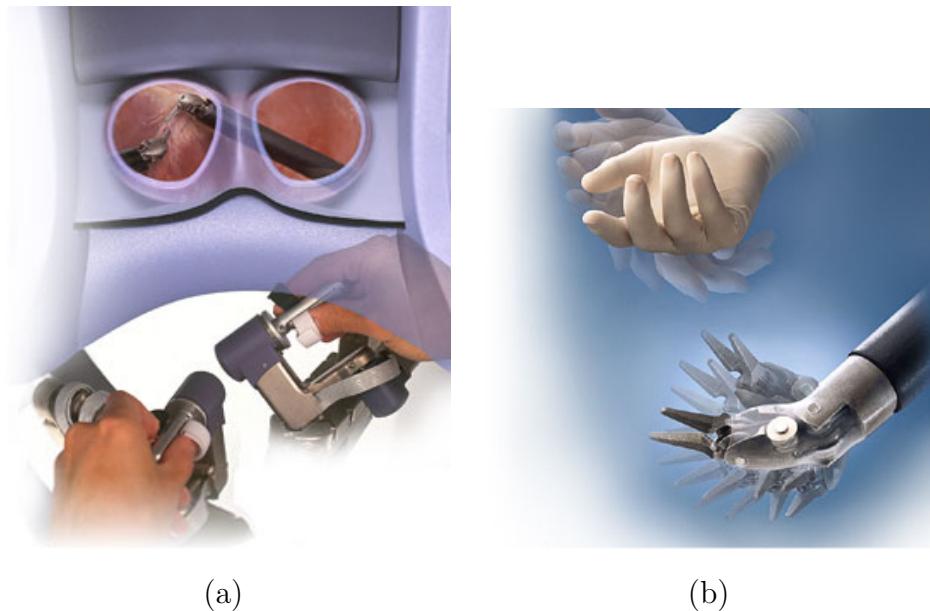


Figura 3.2: Strumentazione chirurgica robotizzata: (a) ditali attuati con feedback di forza. (b) microbisturi che operano in via laparoscopica

aiutare ad apprendere meglio e sul campo (“virtuale”) le tecniche chirurgiche. Sono nati per questo, software di simulazione chirurgica, capaci di simulare veri e propri specifici interventi.

Altri sistemi aptici sono quelli di supporto alla pianificazione di un intervento. Si ricostruisce la scena operatoria utilizzando le informazioni proprie del paziente, acquisite in fase d’indagine (medical imaging), e viene simulato l’intervento, prima che questo venga realmente eseguito sul paziente, in modo da studiare a priori gli effetti post-operatori e le complicazioni che possono verificarsi durante l’intervento.

Infine l’utilizzo delle tecnologie aptiche nei sistemi diagnostici. Per esempio, la possibilità di sentire la consistenza di un tessuto, o, perché no, far sentire ad una mamma i lineamenti del viso del figlio ancora in grembo.

Proprio a questo è finalizzato il lavoro di questa tesi. Si vogliono utilizzare gli strumenti del medical imaging e delle tecnologie aptiche, per realizzare una

CAPITOLO 3. UTILIZZO DELLE INTERFACCE APTICHE IN MEDICINA 55

workstation visio-aptica, capace di generare una sensazione tattile di un feto a partire da un esame ecografico.

Capitolo 4

Progettazione *Software*

Il processo di ricostruzione di isosuperfici affrontato nel Capitolo 1 e l'insieme di tecniche studiate nel Capitolo 2, hanno portato allo sviluppo di due progetti software distinti. Con il primo US3D¹ (Fig 4.1) viene generata la isosuperficie di interesse, che è poi fornita in ingresso al secondo software, US3DTouch, che gestisce l'interazione visio-aptica: operatore \leftrightarrow PHANToM \leftrightarrow isosuperficie. Particolare attenzione è stata data alla scelta dei sistemi di sviluppo ed alle librerie utilizzate. L'obiettivo era quello di ottenere un codice multiplatforma, da poter quindi compilare e eseguire su sistemi grafici come la Silicon Graphics. L'interfaccia utente GUI (Graphics User Interface) è stata sviluppata con le librerie FLTK. Un apporto fondamentale nella progettazione del software è venuto dalle librerie VTK, utilizzate in modo intensivo per la gestione, visualizzazione ed elaborazione di immagini e superfici. Infine l'utilizzo del PHANToM SDK per la gestione dell'interfaccia aptica della Seansable². L'ambiente di sviluppo utilizzato è il Microsoft Visual C++ 6.0 su piattaforma Windows 2000.

¹UltraSound 3D

²Per i dettagli delle librerie fare riferimento al Capitolo 5

4.1 US3D

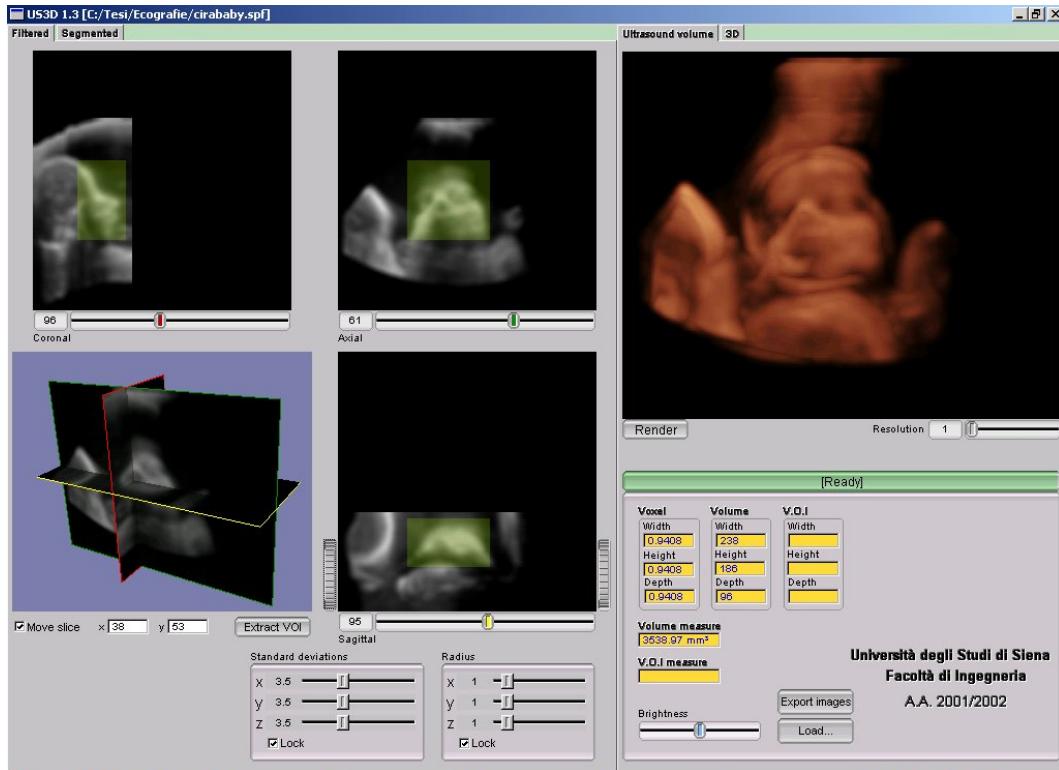


Figura 4.1: Interfaccia utente. In alto a destra si puo vedere la visualizzazione volumetrica diretta

In Fig 4.2 è riportato il pannello dati, dove vengono visualizzate le informazioni circa il volume ecografico in esame.

Per eseguire la fase di *Load* è stata implementata la classe *vtkSPFReader* che si occupa di :

1. leggere l'header del file SPF³
2. eseguire il parsing dei tag, in modo da ottenere le informazioni del volume

³Siemens Project Files: è il file in formato DICOM dove viene memorizzato il volume ecografico



Figura 4.2: Pannello dati

3. leggere la sequenza delle immagini, ed allocarle in memoria centrale

Il volume viene poi visualizzato sui tre piani di vista differenti Axial, Sagittal e Coronal (Fig 4.3), e con tecniche di visualizzazione volumetrica diretta attraverso la classe vtkRayCastMapper(ray-casting) (riquadro in alto a destra in Fig 4.1). E' anche possibile estrarre, dal file SPF, le singole immagini, ed esportarle in file in formato bitmap (BMP) (volume.0, volume.1,...etc), attraverso il comando di Export, per questa operazione viene utilizzata la classe vtkBMPWriter. La vista dei tre piani è ottenuta considerando che $V(i,j,k)$ è il voxel associato al pixel :

- $P(i, j)$ dell' immagine $k - esima$ nel piano *Axial*
- $P(k, j)$ dell' immagine $i - esima$ nel piano *Coronal*
- $P(i, k)$ dell'immagine $j - esima$ nel piano *Sagittal*

Al volume viene quindi applicato il filtro di smoothing gaussiano, con parametri di default : $\sigma_x = \sigma_y = \sigma_z = 3$ che definiscono la forma della gaussiana, e $\rho = 1$ che ne indica l'ampiezza⁴ della finestra di filtraggio. Nella regione (a) della Fig 4.4 vengono visualizzate le tre immagini correnti, delle tre viste, secondo la loro orientazione spaziale, in accordo con la Fig 4.3.

⁴espressa in numero di pixel

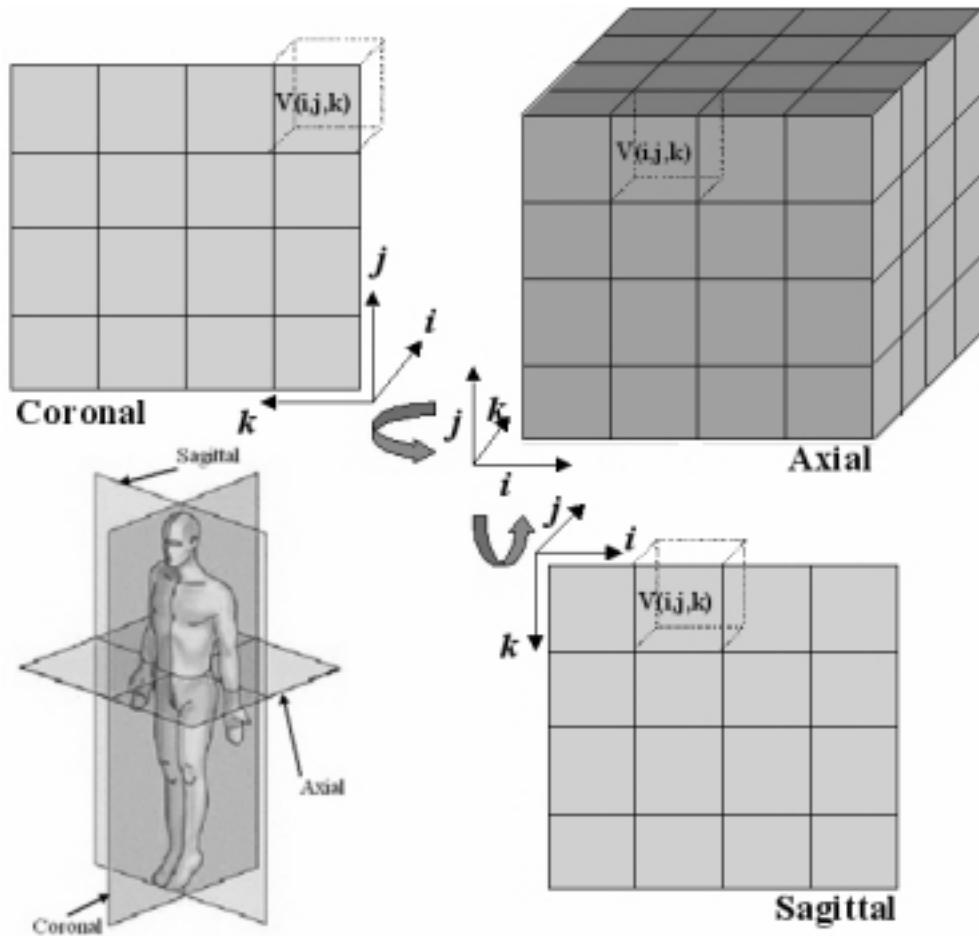


Figura 4.3: Sistemi di riferimento dei tre piani di vista axial, sagittal, coronal.

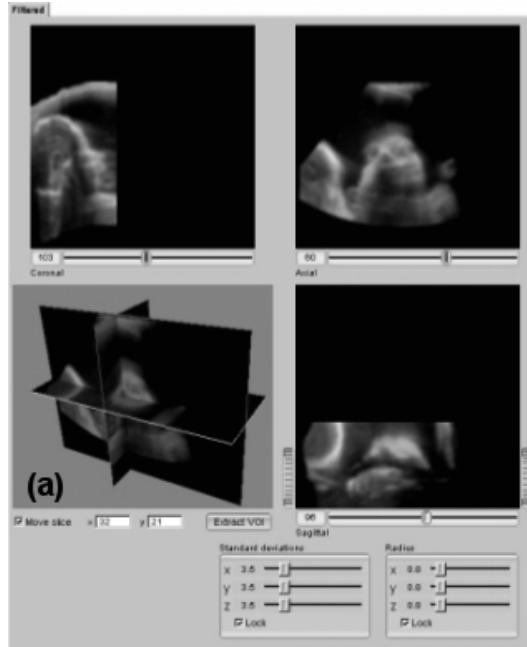


Figura 4.4: Pannello di vista e di filtraggio del volume.

A questo punto si può selezionare un volume di interesse⁵ tracciando con il mouse sulla vista *Axial* un rettangolo, che delimita il volume totale lungo le direzioni i e j . Mentre attraverso i controlli (a) e (b) in Fig 4.5 (a), si delimita il volume lungo k . Infine è sufficiente premere su *ExtractVOI* per concludere l'operazione.

Si passa, quindi, alla fase di segmentazione. Attraverso i controlli *Inf* e *Sup* si seleziona la banda $[T_L, T_H]$ di livelli di grigio da usare per l'operazione di *thresholding*, così da identificare la isosuperficie da estrarre con l'algoritmo del Marching Cubes (Fig 4.5 (b)).

Una volta nota la banda di livelli di grigio $[T_L, T_H]$ che caratterizza la isosuperficie di interesse, questa viene generata semplicemente premendo sul bottone *Render*, che si occupa di fornire in ingresso alla classe *vtkMarchingCubes*, il volume di interesse estratto e l'intervallo $[T_L, T_H]$ (Fig 4.7). Generalmen-

⁵VOI=Volume Of Interest

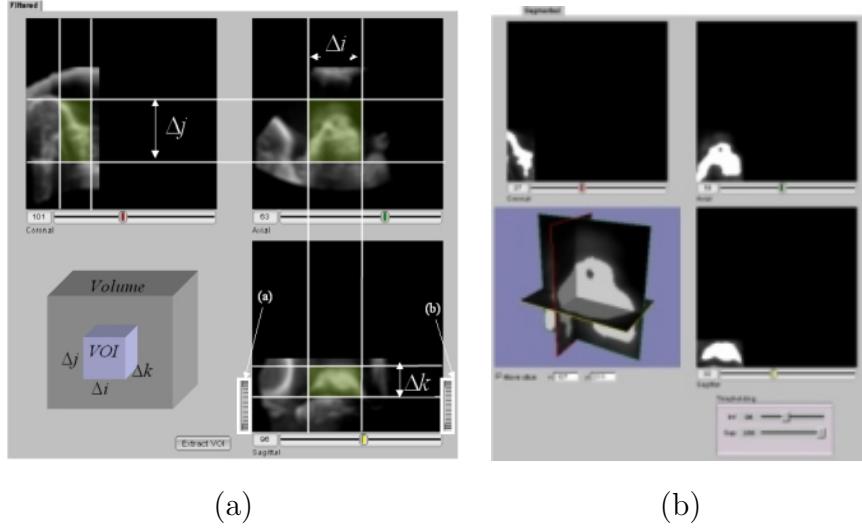


Figura 4.5: (a) Estrazione del volume di interesse. (b) Identificazione della isosuperficie di interesse.

te vengono generate anche una serie di piccole superfici, sconnesse da quella di interesse, causa la non perfetta segmentazione. Per eliminarle viene usato il filtro *vtkPolyDataConnectivity* che estrae dall’insieme di superfici generate, quella connessa più grande. Poi per rendere più liscia la superficie si può agire sul controllo di *Smoothing*. Mentre per ridurne la complessità si utilizza il controllo *Decimate*, dove si seleziona la percentuale di poligoni da eliminare. E’ anche possibile scegliere il tipo di ombreggiatura da usare nel processo di surface rendering [29]:

- flat shading (Fig 4.6 (a)), il nome stesso indica che questo tipo di ombreggiatura produce un effetto di “sfaccettatura” della superficie.
- gouraund shading (Fig 4.6 (b)), viene eseguita una interpolazione delle normali, il risultato ottenuto è senz’altro migliore dando una sensazione di una superficie più liscia.
- wireframe, viene visualizzata la struttura della superficie, quindi, vertici e segmenti.

ed il livello di trasparenza della superficie, attraverso il controllo di *Opacity*.

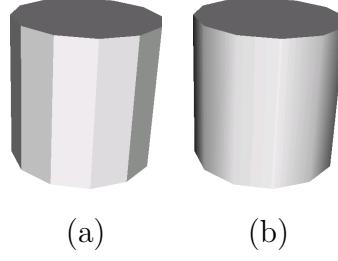


Figura 4.6: Tipi di ombreggiatura usati nel processo di surface rendering: (a) flat shading. (b) gouraund shading

Una volta soddisfatti della superficie generata è possibile salvarla su disco, in formato VRML (per visualizzarla anche con un browser internet) attraverso la classe vtkVRMLExporter, oppure nel più compatto formato binario VTK con la classe vtkPolyDataWriter. Tale superficie sarà poi fornita in ingresso al software *US3DTouch*.

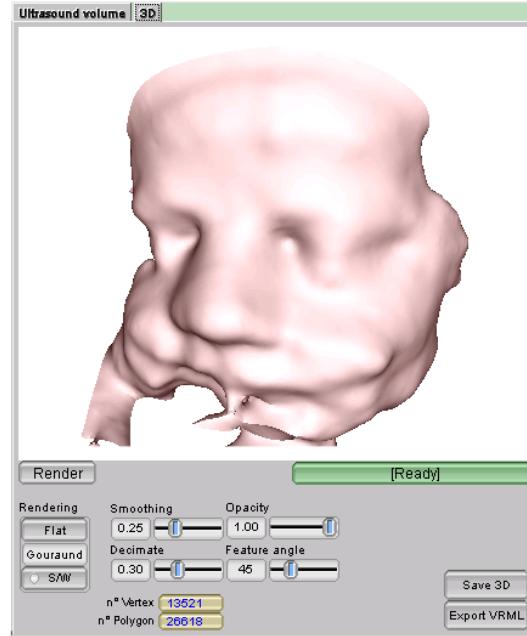


Figura 4.7: Esempio di surface rendering.

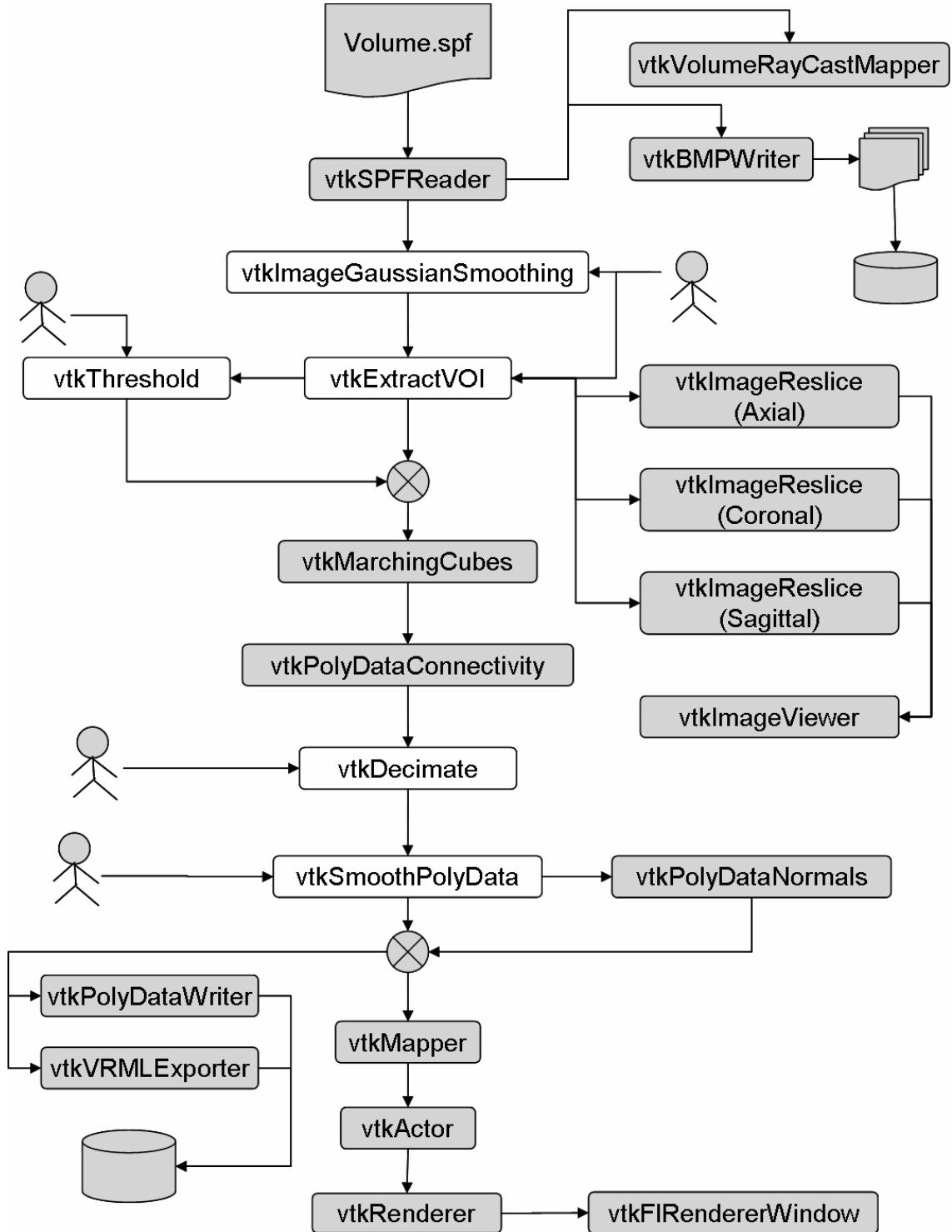


Figura 4.8: Pipeline di ricostruzione 3D

4.2 *US3D Touch*

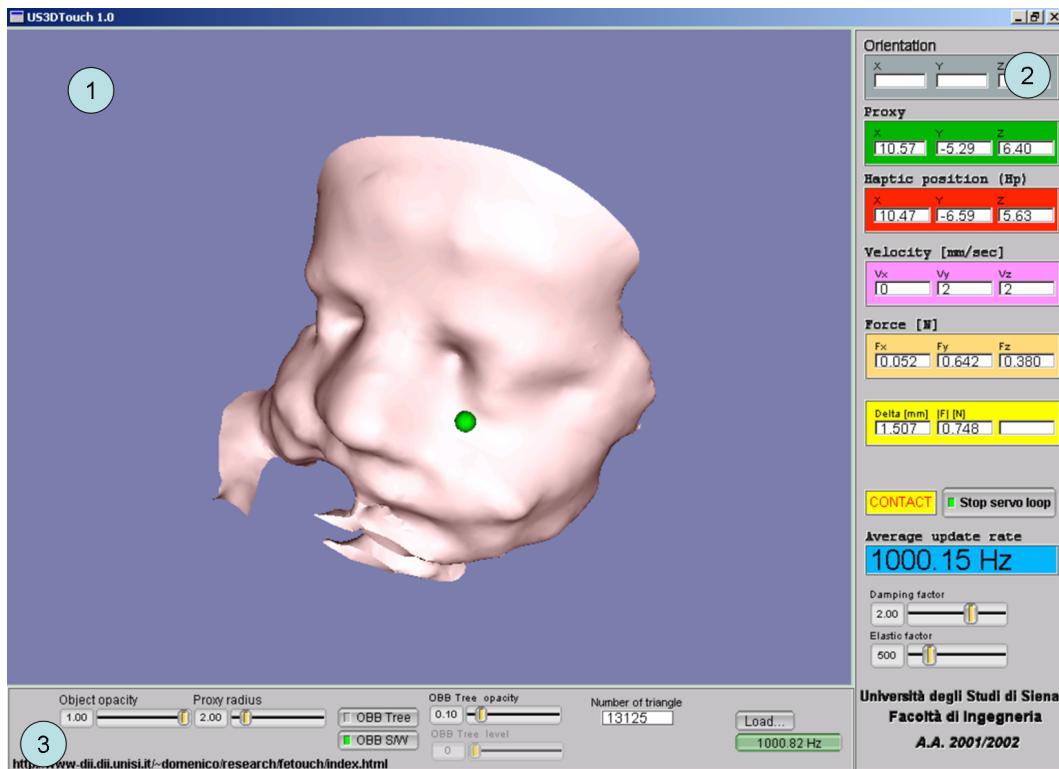


Figura 4.9: Interfaccia utente

Una volta caricato un modello tridimensionale (surface.vtk) memorizzato in formato VTK (vtkPolyDataReader), ne viene automaticamente generato l'OBB-Tree (vtkOBBTree) dove la profondità massima, in termini di livelli dell'albero, è stato fissata a 9, mentre la soglia sul numero massimo di triangoli contenuti in ciascun OBB foglia è pari a 25. Tali scelte sono state dettate dai tempi di risposta ottenuti con il PHANToM (vincolo di 1KHz sul servoLoop()) con modelli di complessità differenti, 10-40.000 triangoli.

L'interfaccia utente è suddivisa in tre zone Fig 4.9. Nella zona **1** viene rappresentata la scena virtuale dove figurano i diversi attori : il modello tridimensionale caricato, il proxy Fig 2.9(rappresentato come una sfera di colore verde),

la posizione corrente del PHANToM (sfera rossa), il sistema molla-smorzatore per il force rendering Fig 2.11 (segmento di colore bianco) ed infine l'OOBTree (di colore giallo), di cui vengono visualizzati gli OBB di ciascun livello Fig 2.5. Il pannello di comandi nella zona **3** serve per agire sugli attori della scena. E' pertanto possibile:

- settare il livello di trasparenza dell'attore principale (oggetto da toccare)
- modificare le dimensioni delle due sfere (rossa e verde)
- visualizzare o no l'OBB-Tree
- scegliere il tipo di visualizzazione degli OBB : solido o strutturale (wire frame)
- selezionare il livello dell'OBB-Tree da visualizzare

Infine il pannello nella zona **2** riporta lo stato del PHANToM:

- posizione
- velocità
- forza
- frequenza di servoLoop(), quest'ultimo si può far partire o fermare con un apposito controllo

e lo stato del sistema molla-smorzatore Pag 2.11, sul quale è possibile agire (off-line o on-line) con appositi controlli, andando a modificare la costante elastica (K) e il fattore di smorzamento (D).

Infine per questioni di sicurezza del PHANToM è stato fissato un livello di soglia sulla forza generata, pari a 5N, rispetto ai 6.4N riportati dalla casa produttrice in Tab 2.1.

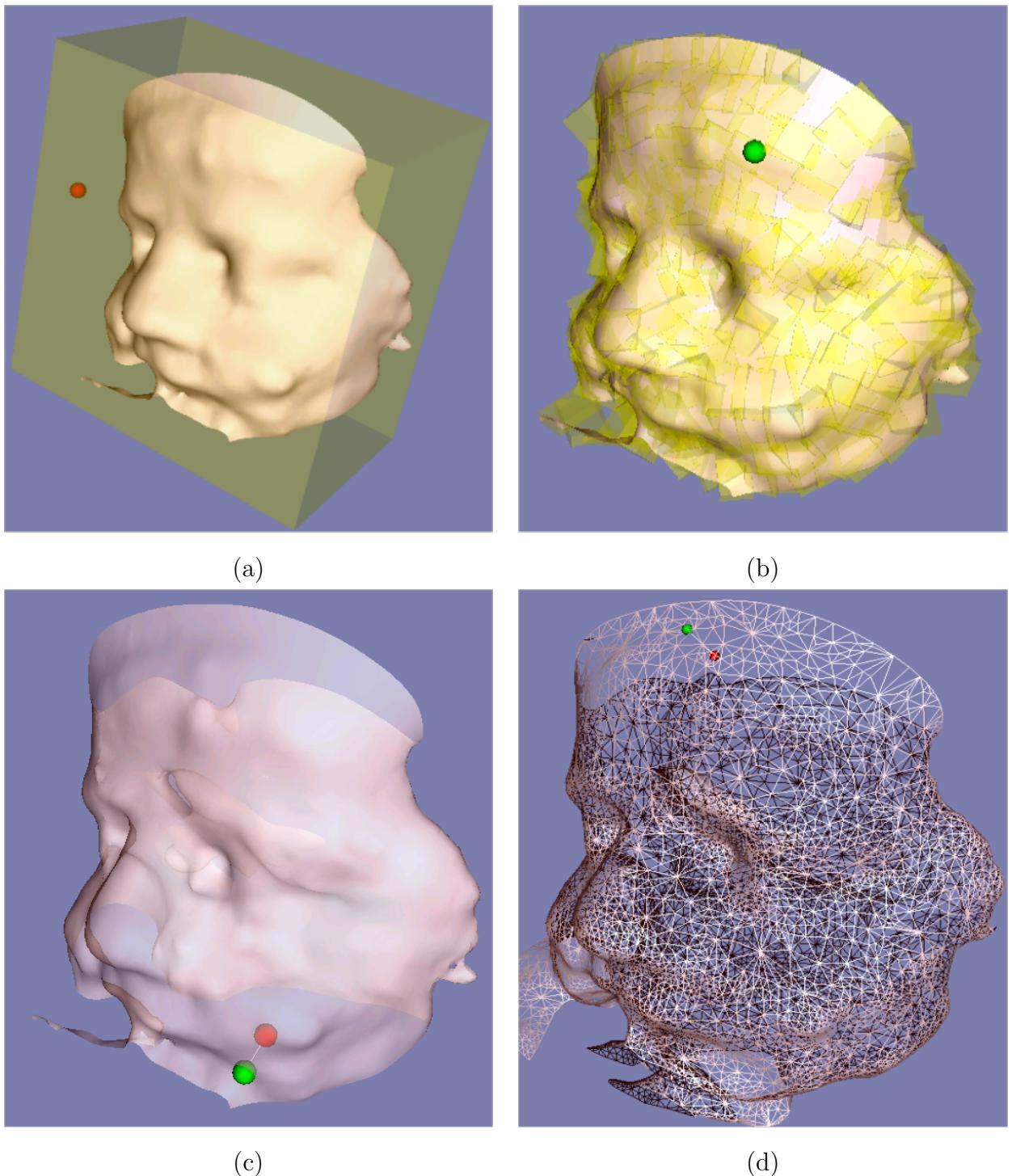


Figura 4.10: (a) OBB-Tree livello 0. (b) OBB-Tree livello 9. (c) Effetto di trasparenza, si vedono il proxy e la posizione del PHANToM. (d) Struttura della superficie, 16.702 triangoli.

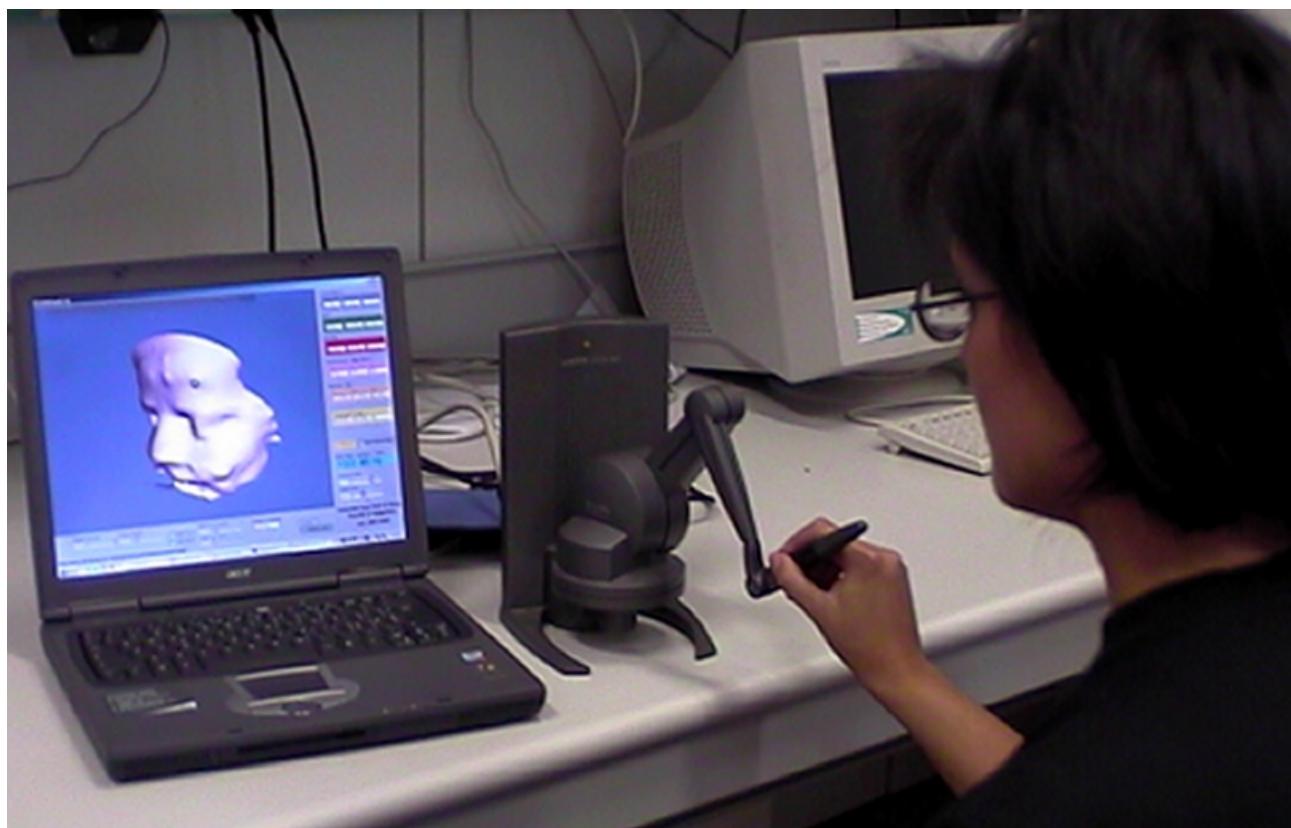


Figura 4.11: Workstation visio-aptica in funzione presso il laboratorio di Automatica e Robotica del Dipartimento di Ingegneria dell'Informazione dell'Università di Siena.

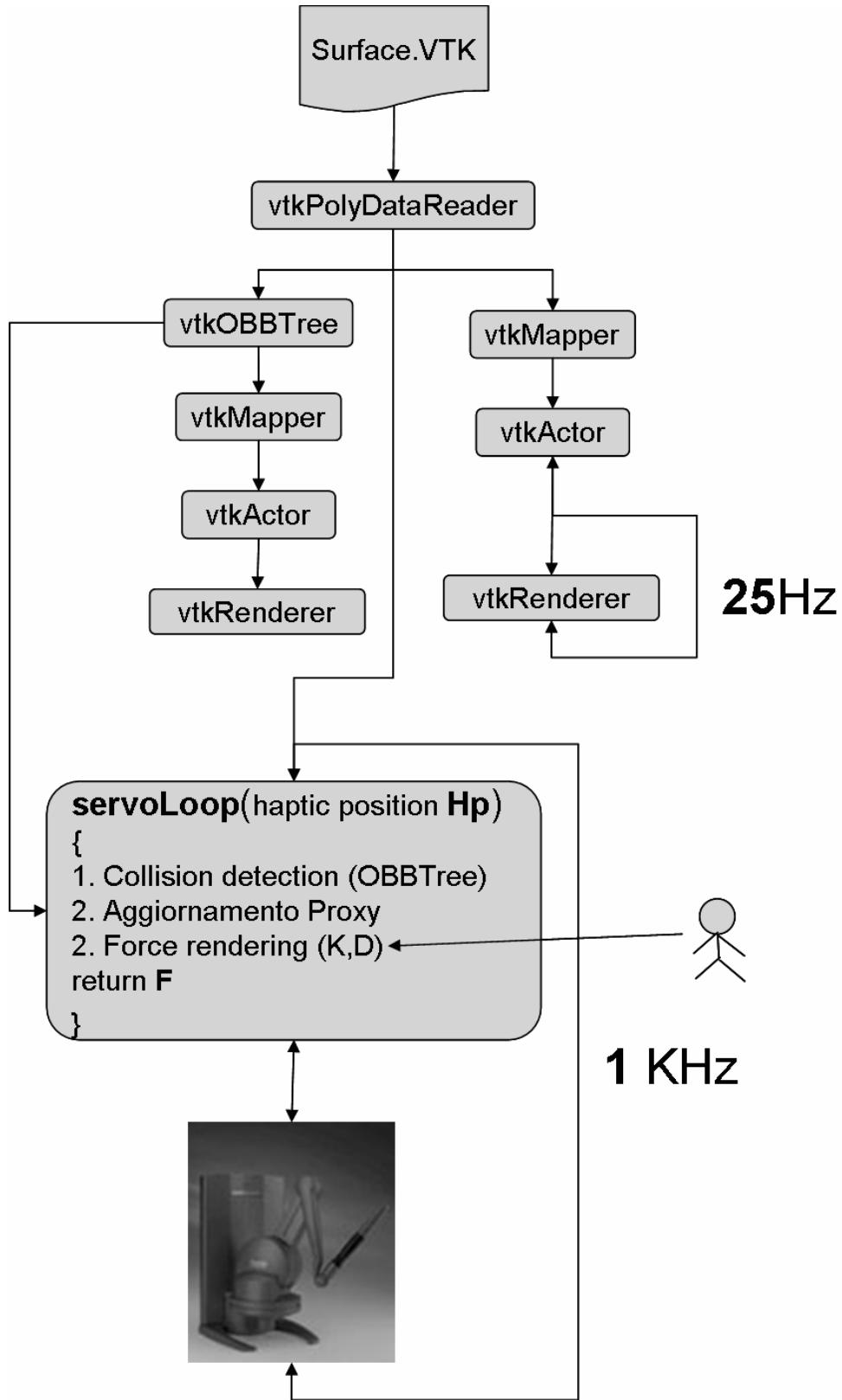


Figura 4.12: Struttura di US3DTOUCH ed interfacciamento con il PHANToM

Capitolo 5

Librerie

5.1 FLTK

Il “Fast Light Tool Kit” [33](FLTK, da pronunciare come *fulltick*) è una libreria ad oggetti (C++), open-source (con licenza LGPL¹), multipiattaforma (Unix®(X11), MacOS®, Microsoft Windows®), orientata allo sviluppo di interfacce utente (GUI). Inizialmente sviluppata da Mr. Bill Spitzak, ora continua ad esserlo da parte di una piccola comunità di programmatore sparsi per il mondo con sede centrale negli US. Le peculiarità delle FLTK che ne hanno decretato il successo sono la velocità e la compattezza, dovute ad un’attenta ottimizzazione del codice, l’alta integrazione con librerie grafiche come OpenGL®e la compatibilità a livello di header con altri toolkit di GUI come GLUT e XForms. Questo rende le FLTK un ottimo strumento di sviluppo per applicazioni grafiche veloci e multipiattaforma. Come ogni libreria sviluppata ad oggetti, è composta da una serie di classi che implementano i differenti elementi grafici (widget) come bottoni, finestre, liste, html browser, scroll bar, etc. La scrittura di un programma è molto semplice ed intuitiva grazie alla standardizzazione dei metodi di ciascun widget. Per esempio con la funzione

¹GNU Lessere General Public License

Fl_Widget → value() è possibile modificare o leggere il contenuto informativo di ciascun widget, come il numero memorizzato in un oggetto contatore.

Vediamo un primo programma scritto con le FLTK, il classico esempio dell'hello world:

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include<FL/Fl_Box.H>

int main(int argc, char **argv)
{
    Fl_Window *window = new Fl_Window(300,180);
    Fl_Box *box = new Fl_Box(20,40,260,100,"Hello,World!");
    box->box(FL_UP_BOX);
    box->labelsize(36);
    box->labelfont(FL_BOLD+FL_ITALIC);
    box->labeltype(FL_SHADOW_LABEL);
    window->end();
    window->show(argc,argv);
    return Fl::run();
}
```

Prima viene creata la finestra principale:

```
Fl_Window *window = new Fl_Window(300,180);
```

alla quale viene aggiunto un controllo box:

```
Fl_Box *box = new Fl_Box(20,40,260,100,"Hello,World!");
```

di cui sono poi settati: il tipo, la dimensione dell'etichetta, il font da usare, e lo stile di visualizzazione del testo.



Figura 5.1: Esempio FLTK : Hello World

```
box->box(FL_UP_BOX);
box->labelsize(36);
box->labelfont(FL_BOLD+FL_ITALIC);
box->labeltype(FL_SHADOW_LABEL);
```

Infine si notifica che è finita la descrizione della finestra principale:

```
window->end();
```

Quindi si visualizza la finestra e si lancia il loop principale di FLTK, necessario per avere le notifiche degli eventi dai diversi widget.

```
window->show(argc,argv);
return Fl::run();
```

Quando si scrivono programmi che utilizzano pochi elementi grafici si può anche procedere in questo modo e scrivere il codice in modo manuale. Il problema nasce, però, quando si scrivono applicativi che fanno uso di molti controlli e soprattutto delle funzioni di callback associate a ciascuno di essi. In questo caso ci viene in aiuto una utility molto efficace e semplice da usare : FLUID (Fast Light User Interface Designer).

Si tratta di un editor grafico, con il quale si possono disegnare le GUI con un approccio WYSIWYG (What You See Is What You Get) e salvarle in formato .fl. FLUID può compilare i file .fl e generare il codice C++ corrispondente in due file .cxx .h. Il file .cxx implementa il codice che genera la finestra, e le funzione di callback, mentre il file .h contiene la dichiarazione della classe. Lo stesso esempio Hello World! poteva essere generato con FLUID Fig 5.3.

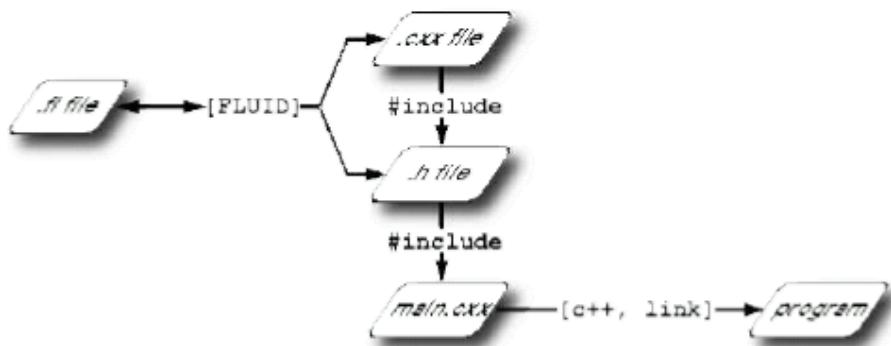


Figura 5.2: Organizzazione di FLUID



Figura 5.3: Esempio FLUID : Hello World

5.2 VTK

5.2.1 Introduzione

Il VTK² [29],vtk] è un software object-oriented open-source, orientato alla computer graphics, visualizzazione ed elaborazione delle immagini. Esso consiste di due sottosistemi di base :

- un insieme di librerie compilate in C++, che costituiscono il cuore del sistema.
- ed un livello più alto, che ne permette l'utilizzo anche con linguaggi interpretati, come : Java, Tcl, Python.

Il codice delle librerie è totalmente multipiattaforma, pertanto è possibile compilare programmi per Windows e le principali piattaforme Unix-like. VTK richiede la presenza di una libreria grafica, come OpenGL, XGL, Mesa o StarBase.

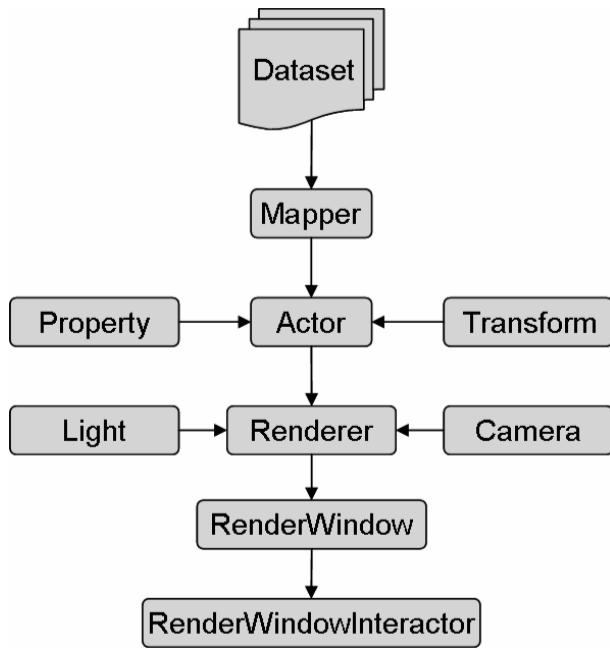
Le classi sono organizzate in due categorie, che si occupano di :

- graphics model
- visualization model

5.2.2 Graphics model

Il ruolo della pipeline grafica è quello di trasformare un insieme di primitive grafiche in immagini. Il paradigma utilizzato per descrivere una scena tridimensionale è quello del set cinematografico, così in VTK troviamo gli oggetti Actor, Light e Camera. Gli attori rappresentano le entità visibili nella scena e sono associati ai mappers che ne forniscono la rappresentazione grafica. Le informazioni circa la posizione, l'orientamento e la scala sono gestite dall'oggetto

²VISualization Toolkit



vtkTransform che incapsula una matrice 4x4, mentre l’aspetto, è controllato dall’oggetto vtkProperty che ne stabilisce le caratteristiche di superficie (colore ambiente, diffuso e speculare, trasparenza) e il tipo di rendering (solido, wireframe). L’oggetto Camera stabilisce in che modo la scena viene proiettata sull’immagine, sono disponibili la proiezione parallela e prospettica, ed è anche possibile abilitare la modalità di resa stereo. Gli attori, le luci, e la camera sono poi passati ad un oggetto Renderer che crea l’immagine della scena. L’oggetto RenderWindow rappresenta una finestra dell’applicativo e può essere suddiviso in più quadranti per visualizzare una o più scene. Infine l’oggetto RenderWindowInteractor fornisce un semplice supporto per l’interazione, tramite il mouse è possibile ruotare, traslare e fare uno zoom sia della vista che dei singoli attori.

5.2.3 Visualization model

La pipeline di visualizzazione si occupa di costruire una rappresentazione geometrica dei dati, che viene poi resa attraverso la pipeline grafica. Il VTK usa

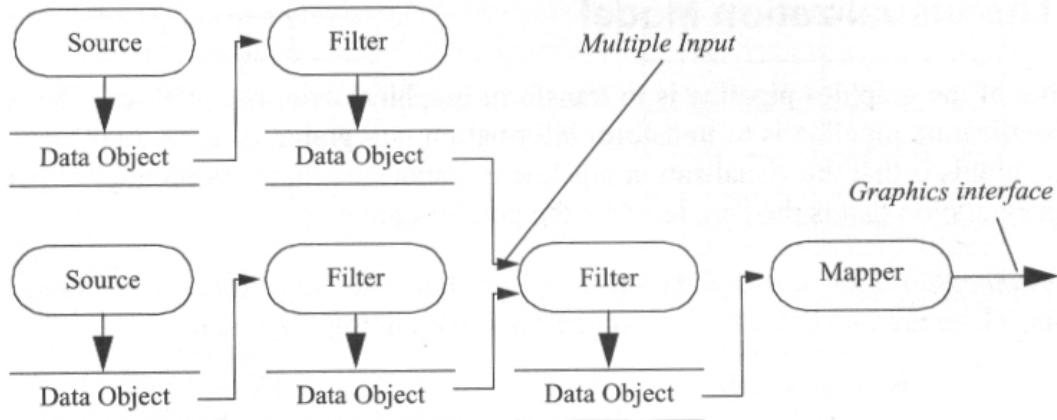
per questo, un approccio data flow che coinvolge due oggetti di base :

1. vtkDataObject
2. vtkProcessObject

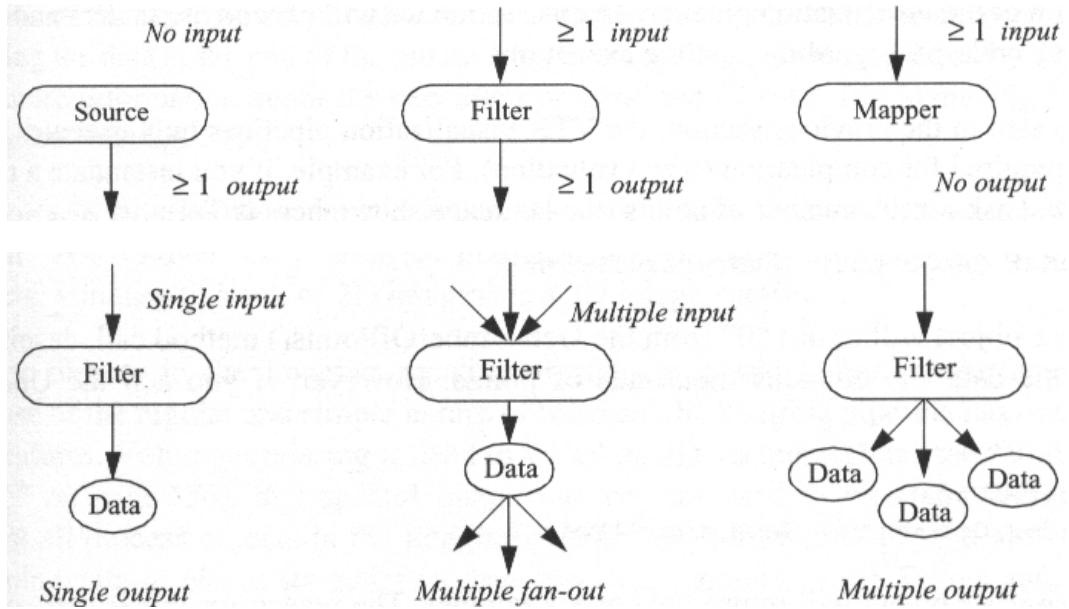
questo perchè nella progettazione delle librerie, si è scelto di separare le classi che contengono i dati, da quelle che gli elaborano. I vtkProcessObject costituiscono, quindi, dei generici filtri che operano sui data objects. Queste due classi di oggetti vengono interconnesse, così da formare la pipeline di visualizzazione (data-flow network). La connessione tra due process object avviene di solito tramite l'istruzione

$$\text{FilterA} \longrightarrow \text{SetInput}(\text{FilterB} \longrightarrow \text{GetOutput}());$$

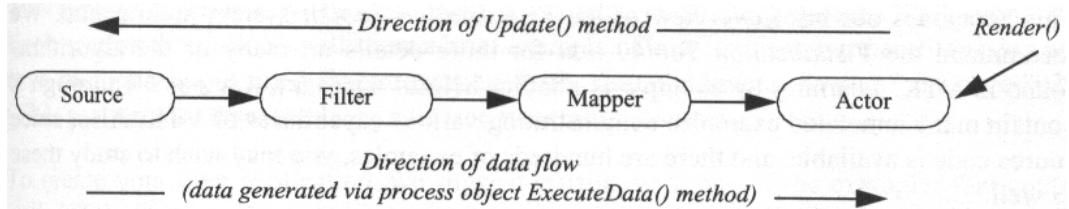
questo è possibile farlo solo se FilterA e FilterB si scambiano un data object di tipo compatibile.



Vi sono poi, diversi tipi di process object. I source process, producono dati, leggendoli da file, o generandone di nuovi. I mappers trasformano i data objects in graphics data che sono poi resi dal renderer. Un esempio di mapper sono i writer, che scrivono su file il contenuto di un data object. Ed infine vi



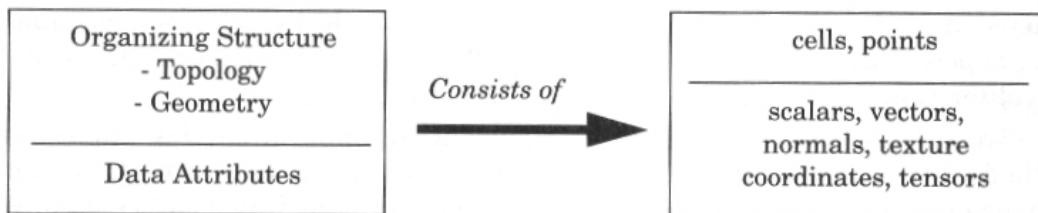
sono i filter che necessariamente hanno un numero di ingressi ed uscite ≥ 1 . Una volta che la data-flow network è stata costruita è necessario mandarla in esecuzione, per avviare l'elaborazione e quindi la visualizzazione dei risultati. Per fare questo nel VTK è stato implementato un meccanismo di sincronizzazione basato su time-stamps. La richiesta di dati in uscita, attraverso il metodo Update, provoca una richiesta a catena a tutti gli altri process object a monte, la pipeline viene così ripercorsa a ritroso fino a trovare il primo elemento utile alla rielaborazione dei dati. Infatti, per evitare elaborazioni inutili, vengono attivati solo quegli oggetti che hanno visto una modifica dei parametri interni oppure dei dati in ingresso.



5.2.4 Data object

I data objects, all'interno della pipeline di visualizzazione sono indicati come *datasets*. La classe (astratta) da cui ogni tipo di dataset concreto è derivato è vtkDataSet, la quale è divisa in due parti :

- una struttura organizzativa
- attributi



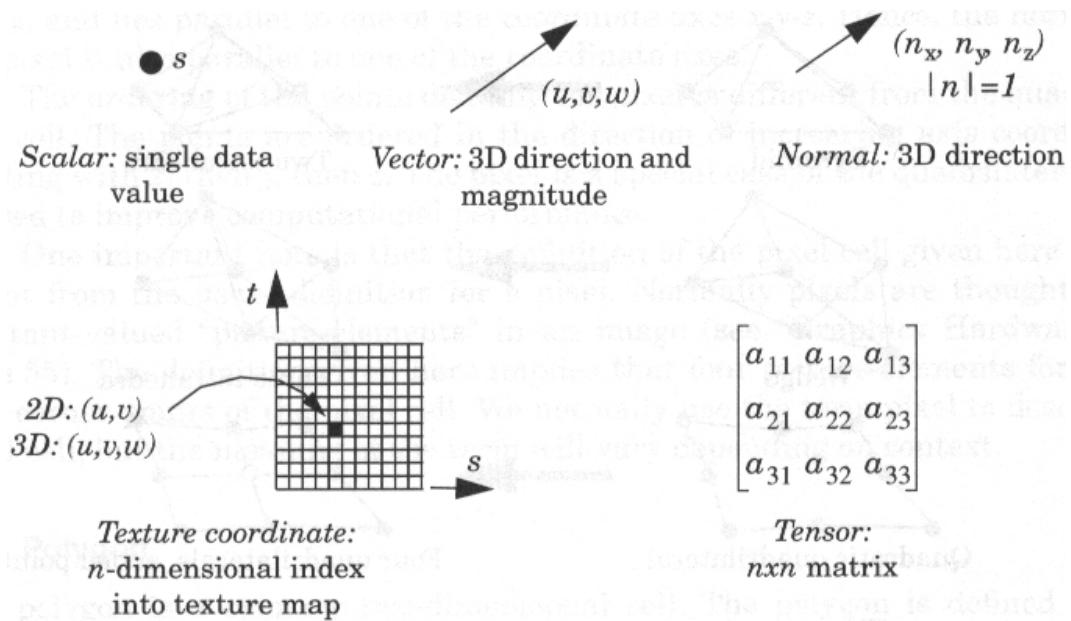
La struttura organizzativa è poi suddivisa in altre due parti :

Topologia : quell'insieme di proprietà che sono invarianti rispetto a certe trasformazioni geometriche: rotazione, traslazione, scala.

Geometria : è una istanza della topologia. Per esempio, dicendo che un poligono è un “triangolo” se ne specifica la topologia. Mentre specificandone le coordinate dei punti, ne stiamo definendo la geometria.

Gli attributi, invece, sono informazioni supplementari associati alla topologia e/o geometria. Per esempio, la temperatura associata a ciascun punto. L'organizzazione del dataset prevede che la struttura sia composta da celle e punti. Dove le celle specificano la topologia, mentre i punti la geometria. Tipicamente gli attributi, consistono in : scalari, vettori, normali, coordinate delle texture, tensori ed altri tipi definiti dall'utente.

Tale tipo di organizzazione deriva dalla semplice osservazione che un insieme di dati per essere elaborato al computer deve essere discretizzato. Quindi, i



punti vengono posizionati dove si conosce il dato, mentre le celle costituiscono una interpolazione del set di dati.³

5.2.5 Tipi di celle

Come detto sopra, un dataset consiste di uno o più celle. Queste costituiscono i blocchi fondamentali per un sistema di visualizzazione. Una cella viene specificata indicandone il tipo, quindi la topologia, cioè il numero dei vertici, dei lati e delle facce, le relazioni di appartenenza e di adiacenza tra queste entità, attraverso un lista ordinata di identificatori di vertici : *connectivity list*. Attraverso questa è poi possibile individuare le coordinate di ciascun vertice nella lista di punti che definisce la geometria della cella : *points list*. I tipi di celle si dividono in elementari e composte, le celle composte sono costituite aggregando celle elementari e sono d'uso comune nelle librerie OpenGL.

L'insieme dei punti e degli identificatori di tutte le celle presenti nella scena sono organizzate in due unici vettori, in modo da ottimizzare l'uso della

³cioè rappresentano i dati tra due passi di campionamento successivi

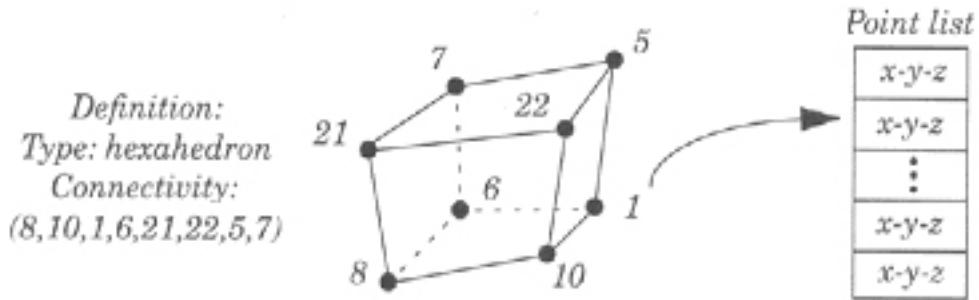


Figura 5.4: Esempio di definizione di una cella

memoria, evitando la presenza di più istanze dello stesso punto. L'insieme delle celle è poi organizzato in un array, `vtkCellTypes`, in cui ogni cella è rappresentata da una coppia tipo-cellula e posizione degli identificatori dei vertici all'interno della connectivity list (`vtkCellArray`). Quadrilateral e Pixel sono topologicamente equivalenti ma il secondo tipo deve avere i lati paralleli agli assi di riferimento e quelli adiacenti perpendicolari tra di loro. Analoghe differenze esistono tra i tipi Hexahedron e Voxel. Va notato che i tipi Pixel e Voxel corrispondono alle definizioni convenzionali di constant valued picture element e volume element solo nel caso in cui gli attributi sono assegnati alle celle.

5.2.6 Tipi di DataSet

Un dataset è una struttura organizzata composta da punti e celle, e la tipologia è definita da come questi sono organizzati. Un dataset può avere quindi una struttura regolare o irregolare. È regolare (strutturata) quando sussiste un'unica relazione matematica che descrive la distribuzione spaziale di punti e celle. Questo tipo di organizzazione è la più efficiente in termini di memoria, in quanto non è necessario memorizzare le coordinate di tutti i punti, perché è possibile calcolarle una volta note le caratteristiche della struttura del dataset. Come il numero di punti lungo le tre direzioni (dimension), la spaziatura tra questi (spacing), e l'origine (origin). Mentre i dataset irregolari (non struttu-

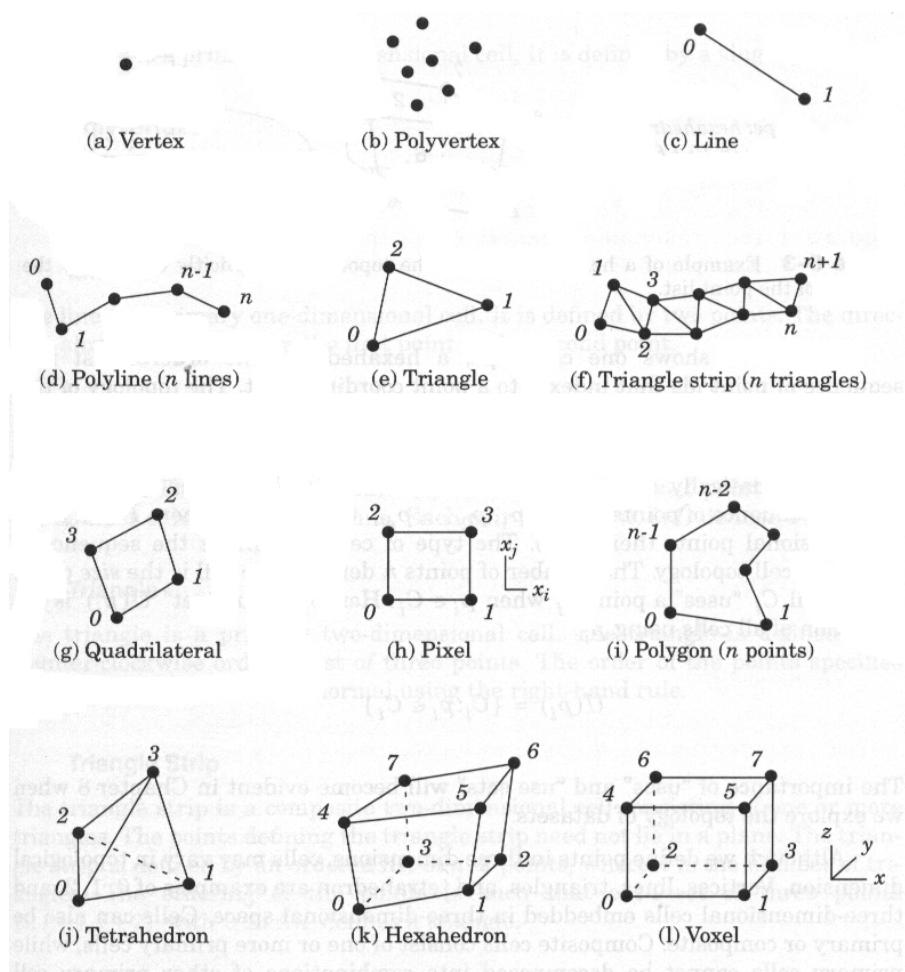
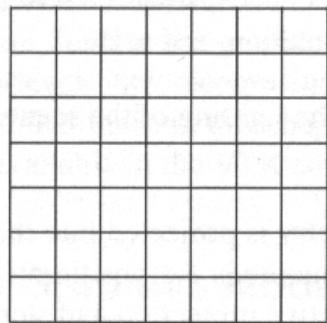
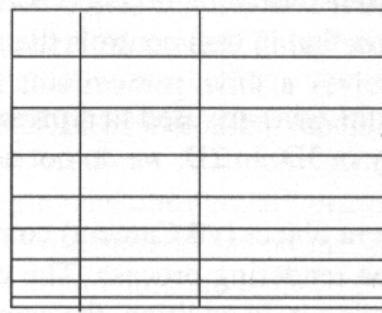


Figura 5.5: Tipi di celle

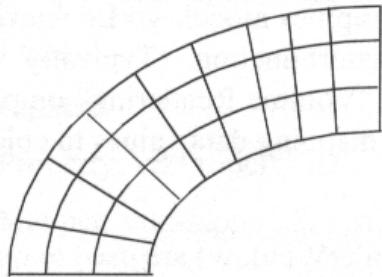
rati) essendo di natura più flessibili e generali necessitano di una rappresentazione esplicita per ogni punto. Due dei tipi più interessanti sono : polygonal data, structured points. Il primo è costituito da primitive geometriche come vertici, linee, triangoli, e si presta bene per gestire la isosuperficie generata dal marching cubes e nel processo di rendering grafico, in quanto le workstation grafiche sono altamente ottimizzate per processare tali tipi di dati. La classe di riferimento è **vtkPolyData**. Il secondo è costituito da un insieme di punti e celle organizzate in modo regolare. Le righe e colonne sono parallele al sistema di riferimento globale. Se l'insieme dei dati è organizzato su un piano, si fa riferimento ad esso come immagine. Se invece è organizzato su un insieme di piani paralleli si fa riferimento ad un volume. La classe **vtkImageData** è pensata per lavorare su immagini o volumi.



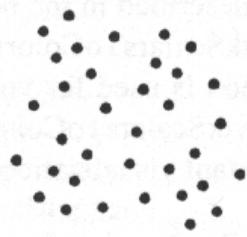
(a) Image Data
(vtkImageData)



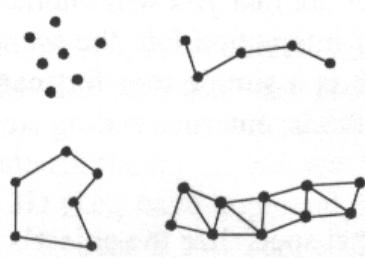
(b) Rectilinear Grid
(vtkRectilinearGrid)



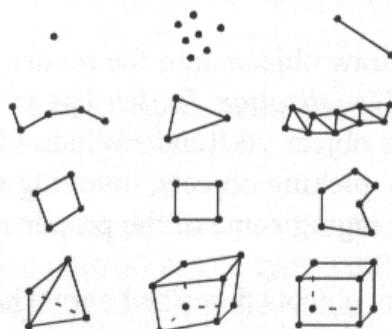
(c) Structured Grid
(vtkStructuredGrid)



(d) Unstructured Points
(use vtkPolyData)



(e) Polygonal Data
(vtkPolyData)



(f) Unstructured Grid
(vtkUnstructuredGrid)

Figura 5.6: Tipi di dataset

5.3 GHOST SDK

Il GHOST SDK⁴ [35, 34] è la libreria sviluppata in C++ dalla SensAble utile alla realizzazione di sistemi virtuali tridimensionali interattivi, che utilizzano i diversi modelli del PHANToM: Desktop, Premium A, Premium T, Premium 6 DOF. I requisiti di sistema richiesti per utilizzare il GHOST SDK sono :

- PC con processore Intel e Windows NT 4.0 (con Service Pack 6) o Windows 2000 (con Service Pack 1), Silicon Graphics Incorporated (SGI) workstations con IRIX 6.5 (Indigo2, Octane, Onyx2.)
- compilatore Microsoft Visual C++ 6.0 o SGI 7.2.1 C++.
- installazione dei PHANTOM Device Drivers v3.1.

La libreria consente di gestire una scena virtuale costituita da un insieme di oggetti, che vengono organizzati in modo gerarchico (albero). I nodi interni rappresentano punti chiavi della scena, e servono per raggruppare logicamente un insieme di nodi che costituiscono un sottoalbero (viene usato per questo la classe `gstSeparator`). Gli elementi appartenenti a tali sottoalberi seguono le trasformazioni (rotazione, traslazione, scala) e la dinamica del nodo chiave. Mentre i nodi foglia dell'albero rappresentano le entità geometriche presenti nella scena Fig 5.7. Il GHOST SDK gestisce la scena solo dal punto di vista aptico non è prevista pertanto la gestione grafica. Ma per fare questo il programmatore è libero di scegliersi un qualsiasi motore grafico, anche se si consiglia l'uso di librerie già collaudate con successo come OpenGL o Open Inventor.

L'insieme delle classi definite nel GHOST SDK sono molteplici⁵ ma di nostro interesse ne sono solo poche, visto che nello sviluppo di US3DTouch

⁴General Haptics Open Software Toolkit Software Developer's Toolkit

⁵si rimanda per una descrizione accurata di tutte le classi alla documentazione del GHOST SDK [34]

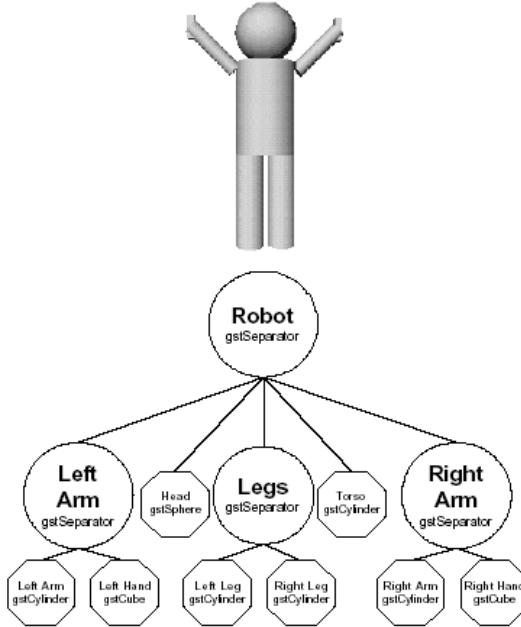


Figura 5.7: Esempio di definizione di un'oggetto

non sono stati utilizzati in alcun modo gli strumenti di collision detection e force rendering offerti dalla libreria.

In particolare si è fatto uso delle sole classi:

gstForceField: che permette di mandare in modo diretto un vettore di forza al PHANToM, attraverso il metodo :

```
virtual gstVector calculateForceFieldForce(gstPHANToM* PHANToM );
```

che fa le veci della funzione di servoLoop() vista in Fig 4.12

gstPHANToM: che è rappresentativo del PHANToM nella scena, permette di accedere allo stato del dispositivo aptico, quindi, posizione, velocità, frequenza di aggiornamento etc. Per fare questo vi sono definiti una serie di metodi:

```

virtual gstPoint getPosition_WC();
const gstVector& getVelocity() const;
double getAverageUpdateRate();

```

gstSeparator rappresenta un nodo chiave dell’albero, a cui si possono aggiungere dei figli per mezzo del metodo:

```
virtual void addChild(gstTransform* newChild);
```

Quindi la scena minima da descrivere per realizzare un processo aptico deve essere costituita da un campo di forza (`gstForceField`) e dal PHANToM (`gstPHANToM`), in Fig 5.8 si può vedere la semplice gerarchia utilizzata nel software US3DTouch.

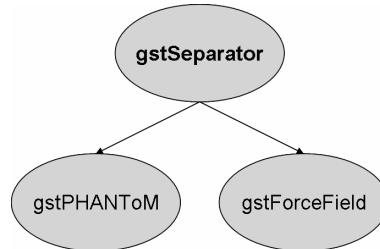


Figura 5.8: Struttura gerarchica della scena

Infine c’è da dire che, dato che sono stati solo utilizzati gli strumenti offerti dal GHOST SDK per recuperare informazioni sullo stato del PHANToM ed attuare un vettore di forza, si può pensare di estendere con facilità l’utilizzo di US3DTouch anche ad altri dispositivi aptici in commercio.

Conclusioni e sviluppi futuri

In questa tesi sono stati studiati alcuni degli strumenti offerti dal medical imaging e delle diverse problematiche inerenti le interfacce aptiche. L'insieme di queste tecnologie ha permesso lo sviluppo di una workstation visio-aptica capace di generare, a partire da un esame ecografico, una percezione visiva e tattile di un feto. Notevoli sviluppi si prospettano in questa direzione. Il passo successivo è quello di generare una sensazione aptica del battito cardiaco del feto.

Un altro possibile sviluppo di questa tesi consiste nello studio di tecniche avanzate per la ricostruzione tridimensionale, a partire da immagini ecografiche. In particolare si vuole rilassare, per quanto possibile, l'ipotesi di piani paralleli nella fase di acquisizione attraverso strumenti ecografici sprovvisti per la sonda di sensori di localizzazione spaziale. Altro punto d'interesse è quello di poter rielaborare una vista del volume lungo un qualsiasi piano di sezione, prima di eseguire le tre viste axial, coronal, sagittal. Questo per migliorare la fase di selezione della regione di interesse.

Altro obiettivo è quello di modellare le superfici, estratte dai volumi ecografici, come oggetti cedevoli. Per fare questo è necessario definire un modello fisico, sul modello geometrico, che descriva l'evoluzione dinamica della superficie, in presenza di un campo di forza. Si potrebbe pensare di usare una modellizzazione agli elementi finiti, ma questo è molto dispendioso in termini computazionali. Un approccio più semplice è invece quello di modellare la

superficie come una rete costituita da masse, molle e smorzatori. Nello specifico viene attribuito a ciascun vertice della superficie una massa ed a ciascun segmento congiungente i vertici un sistema molla-smorzatore. In questo modo quando viene esercitata una forza su un punto della superficie, viene modificato lo stato della rete che quindi produce un diversa disposizione spaziale dei vertici, cioè una deformazione della superficie. Anche in questo caso non è pensabile poter elaborare tutto il modello dinamico (il numero di variabili di stato è proibitivo), ma è senz'altro preferibile seguire un approccio di tipo locale.

Infine si è pensato di trasferire il codice relativo alla gestione della GUI, da FLTK alle più professionali Qt [37], che sono delle librerie commerciali multipiattaforma, ma delle quali è anche disponibile una versione freeware liberamente utilizzabile per lo sviluppo di software non commerciale. L'utilizzo delle Qt può garantire senz'altro una maggiore stabilità del software ed in più sono anche disponibili delle classi che permettono di interfacciare il VTK con le Qt.

Appendice A

CD-ROM

A.1 Contenuto

/Documenti/ articoli e documentazione della bibliografia.

/Ecografie/ alcune ecografie d'esempio da usare con US3D.

/Modelli 3D/ modelli 3D in formato vtk da usare con US3DTouch.

/Software/US3D/ codice sorgente e nella cartella /US3D/Release/ vi è il file eseguibile.

/Software/US3DTouch/ codice sorgente e nella cartella /US3DTouch/Release/ vi è il file eseguibile.

/FLTK/ file fltk.zip costituito da:

fltk-1.1.0b12-source.zip sorgente della libreria

/FLTK/fltk-1.1.0b12-source/ sorgente compilato, quindi librerie statiche (lib) e dinamiche (dll).

fltk.pdf manuale

/VTK/ file vtk.zip costituito da:

/vtk40Src/ sorgente della libreria

/VTK/ sorgente compilato, quindi librerie statiche (lib) e dinamiche (dll)

vtk40Doc.chm documentazione della libreria

vtk40Data.zip dati utilizzati dai programmi di esempio

vtk40Cpp.exe autoinstallante degli header della libreria.

vtk40Core.exe autoinstallante delle dll della libreria.

/PHANToM/ vi sono i driver del PHANToM ed il GHOST SDK.

A.2 Compilazione del codice

Per prima cosa è necessario installare le librerie. Per il VTK bisogna installare vtk40Cpp.exe e vtk40Core.exe. E' da precisare che questi due file non installano tre file necessari che è pertanto necessario copiare dalla directory /VTK/VTK/ e che sono : vtkMarchingCubes.h,vtkPatented.lib,vtkPatented.dll. Per l'FLTK quanto serve è presente nella directory /FLTK/fltk-1.1.0b12-source/ basta copiare tale directory sull'hard disk. Ed infine per il GHOST SDK è necessario prima installare i driver del PHANToM e poi l'SDK. Una volta installate le librerie è necessario configurare il Microsoft Visual C++ v6.0. In particolare bisogna settare i path degli include (.h) e delle librerie (.lib). Selezionare la voce del menu **Tool→Options→Directories** (Fig A.1). I path generalmente sono :

Library

FLTK

C:\Tesi\Librerie\FLTK\fltk-1.1.0b12-source\fltk-1.1.0b12\lib

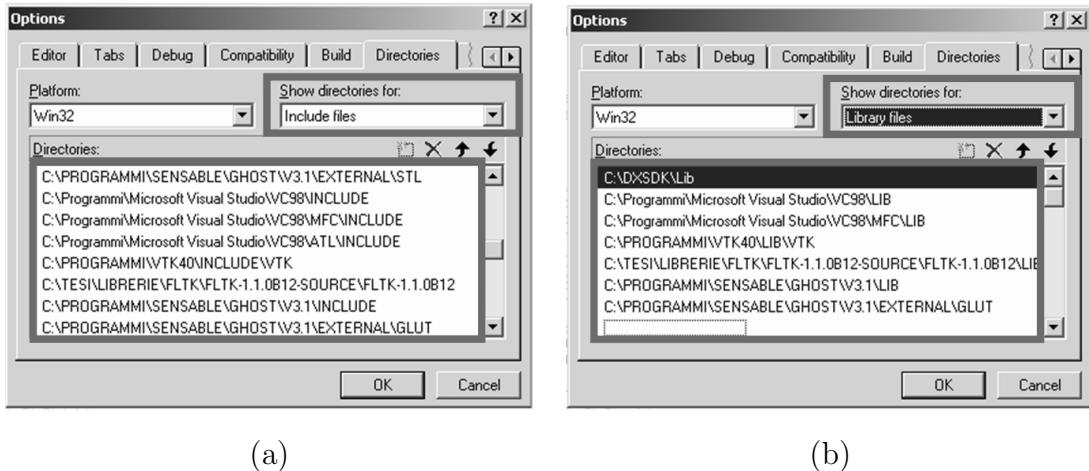


Figura A.1: (a) settaggio dei path dei file di include (.h). (b) settaggio dei path dei file di library (.lib).

VTK

C:\PROGRAMMI\VTK40\LIB\VTK

GHOST SDK

C:\PROGRAMMI\SENSABLE\GHOST\V3.1\LIB

C:\PROGRAMMI\SENSABLE\GHOST\V3.1\EXTERNAL\GLUT

Include

FLTK

C:\TESI\LIBRERIE\FLTK\FLTK-1.1.0B12-SOURCE\FLTK-1.1.0B12 VTK

C:\PROGRAMMI\VTK40\INCLUDE\VTK (copiare vtkMarchingCubes.h) GHOST

SDK C:\PROGRAMMI\SENSABLE\GHOST\V3.1\EXTERNAL\STL

C:\PROGRAMMI\SENSABLE\GHOST\V3.1\INCLUDE

C:\PROGRAMMI\SENSABLE\GHOST\V3.1\EXTERNAL\GLUT\

Infine è necessario aggiungere le voci

FL_DLL, vtkIO_EXPORTS, vtk_EXPORTS

in **Project→Settings→C/C++→Preprocessor definitions**, e il nome di tutte le librerie usate in **Project→Settings→Link→Object/Library modules** che sono:

```
fltkdll.lib  
vtkRendering.lib  
vtkGraphics.lib  
vtkImaging.lib  
vtkIO.lib  
vtkFiltering.lib  
vtkCommon.lib  
vtkPatented.lib  
fltkgl.lib  
glut32.lib  
glu32.lib  
opengl32.lib  
GHOST31.lib  
ghostGLUTManager.lib  
ghostGLManager.lib
```

Appendice B

Articolo

Di seguito viene allegato l'articolo “The FeTouch Project” sottomesso alla IEEE Int. Conference of Robotic and Automation.

The Fetouch Project

B. La Torre, D. Prattichizzo*, F. Barbagli, A. Vicino

Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy.

Abstract: Ultrasound technologies have been widely used in gynecology and obstetrics. Modern ultrasound systems allow the reconstruction of a 3D model of the subject being scanned. Even though visual interfaces have reached very high standards, the problem of representing a 3D image on a 2D computer screen still exists. Moreover no physical interaction is possible with such model. The Fetouch system, developed at Siena University in the last two years, partially solves such issues by using stereo visual feedback and haptic devices. While the system can be used with any 3D model obtained from ultrasound scans, its current prime use is to allow mothers to interact with a model of the fetus they are carrying. The system, which is freely available on the project web page, has been tested on twelve cases which have been monitored by doctors at Siena Hospital.

1 Introduction

In the last twenty years ultrasound techniques have grown in popularity among the gynecology and obstetrics communities [2, 19]. Ultrasound technologies have become a standard in detecting several morphologic and functional alterations involving both fetus and internal female genitalia. The success of ultrasonography is mainly due to its non invasive nature, low cost and ease of use.

Medical ultrasound imaging is inherently tomographic, i.e. it provides all the information necessary for the 3D reconstruction.

*Domenico Prattichizzo is with Dip. di Ingegneria dell'Informazione (via Roma, 56 – 53100 Siena), Università di Siena. EMail: prattichizzo@ing.unisi.it, URL: <http://www-dii.ing.unisi.it/prattichizzo>. Phone: +39-0577-234609.



Figure 1: The Fetouch workstation.

Ultrasound machines are based on the same basic principle: ultrasound pulses are sent to the part of the body being scanned and echoes are received. The time delay of the echoes and their intensity allow to create a 2D image of a cross section of the body commonly referred to as the 2D B-scan of the scan plane. However, various types of ultrasound machines exist. Low-cost solutions, normally referred to as freehand 3D systems, are based on small hand-held probes enhanced with a position sensor¹. The 3D ultrasound process, consists of three stages: scanning, reconstruction and visualization as described in [15]. More expensive solutions, normally referred to as real-time three-dimensional (4D) ultrasound imaging technologies, are normally characterized by arrays of 2D transducer which allow them to directly acquire the volume of the part under investigation.

The former systems are often less accurate. Ac-

¹Most common position sensors are electromagnetic, acoustic or optical [7]

quisition errors are typically due to errors in tracking the probe's exact location. In order to limit such errors the reconstruction process, i.e. retrieving 3D data volumes from a series of 2D B-scans, becomes critical [16, 19]. The latter allow the acquisition of an entire volume at each sample and therefore do not need any interpolation process.

The visualization process is normally based on rendering the 3D volume on a standard 2D PC monitor. While this process has proven quite effective, it remains somewhat limited. Depth information is partially lost. Furthermore no physical interaction is allowed. One of the possible ways to enrich the fruition of 3D volumes, one proposed in this paper, is based on the use of haptic devices. Haptic devices are small robotic structures that allow users to touch virtual objects. This is accomplished by measuring the user position, translate such position to a virtual environment, compute collisions and interaction forces between user and virtual objects and then return such forces to the user through the device.

Haptic devices are now widely used in the field of medical simulation for training purposes [4, 6]. Haptic devices applied to medical imaging is also a growing field of research. In [24], for instance, the authors propose a visio-haptic display of 3D angiograms. In [23] force feedback is used to feel edges of 2D ultrasound images. In [1] the authors propose techniques to add force feedback to the display of volume images. The proposed haptic rendering techniques are however based on voxels and force fields, which have been proven to have problems in various situations [25].

2 System description

The system proposed in this paper allows users to reconstruct 3D visual-haptic models from sets of 2D slices obtained using ultrasound machines. Such models can then be touched using any haptic device. While the system has been mainly developed for the case of interaction with fetal models, the scope of the project is wider. In order to make the proposed system as general purpose as possible, the Fetouch workstation has been designed to process data from

standard 2D ultrasound scans in DICOM format [13]. The system can thus be used in conjunction with any ultrasound machine. The system has been successfully tested on the fetuses of twelve women at Siena Hospital in 2001/02. The data presented was acquired using a Siemens Sonoline Elegra system².

In recent times Novint Technologies has announced the release (at the end of 2002) of a commercial product, the e-Touch Sono, which allows users to interact with 3D fetal models [?]. While the idea is similar to the one proposed in this paper, differences exist. For instance, the Novint product will be based on a dedicated 4D ultrasound system. This will certainly ensure a high level of quality but will limit the application. By using images in the DICOM standard, the Fetouch system can be used with data obtained using any ultrasound machinery.

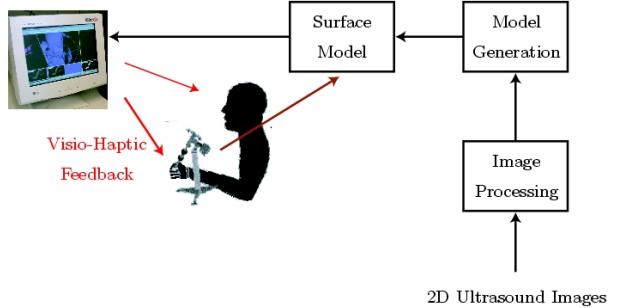


Figure 2: The functional scheme of Fetouch .

It is important to note that the Fetouch system has not been designed with medical diagnosis in gynecology and obstetrics as a prime focus. The user interacts with the surface of a 3D fetal model (or other model). While such surface is enhanced with various effects, such as compliance, heart beat and skin texture, it is important to note that none of these effects are physically based on the data obtained from the ultrasound machinery. While ultrasound data contains information about tissues properties, and such information can be used to simulate different hap-

²Because of a lack of a probe tracking system the scans have been performed following linear trajectories at a constant speed. A scan speed indicator is used to assure such conditions.

tic properties for the system at hand (as for instance in [1]), a more precise system that links such data to the parameters characterizing a deformable fetal model would be needed. Such will be matter of future investigation. The Fetouch system is freely available at [22].

3 System software architecture

The functional scheme of the Fetouch system is reported in Fig. 2. The system is divided in two main blocks serving different functions. The first block (US3D) is devoted to creating a 3D visual-haptic model give a set of ultrasound scans. The second block (US3Dt) allows the user to interact with such system using a haptic device (PHANTOM [21] or Delta [5]) and a 3D image (PC screen alone or enhanced by stereo glasses).

Software has been designed in C++ in an object oriented setting and is portable on various platforms (e.g. Windows and Linux). The Visualization Toolkit (VTK) has been used [18] to create a visual feedback to the user as well as for performing collision detection between the user and the 3D fetal model. The Graphical User Interface has been developed with the fast light toolkit (fltk) [20]. In the following we will focus our attention on the two main blocks that make up the system.

3.1 Automatic model extraction algorithm

This section describes the software for automatic model extraction referred to as US3D. US3D allows ultrasound 2D-scans, in DICOM format, to be gathered and displayed as a volume. In order to better visualize the ultrasound volume, data is re-sliced along three directions (axial, sagittal and coronal) as shown in Fig. 3.

A direct visualization of the ultrasound volume is available in US3D. The Maximum Intensity Projection (MIP) approach is used to render the image in Fig. 4. This technique of direct volume rendering, also known as ray-casting, is based on drawing parallel rays from each pixel of the projection

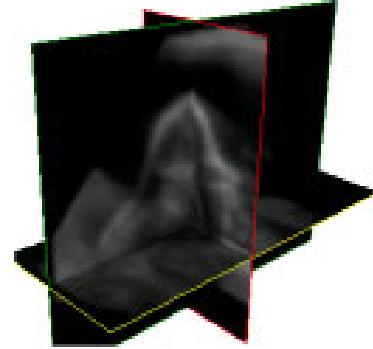


Figure 3: The axial, sagittal and coronal visualization of the fetus.



Figure 4: Volume visualization by maximum intensity projection.

screen and then considering the maximum intensity value encountered along the projection ray for each pixel [10, 11]. This method does not need any pre-processing phase and is fast but it is not truly a 3D visualization technique since any information on depth is lost. The noise affecting raw data can be filtered by a 3D Gaussian smoothing kernel. The volume of interest (VOI) can be selected using the GUI of the US3D software, see Fig. 5.

The VOI is first segmented from the background to obtain, by means of a threshold filter, a binary volumetric data set. The surface fitting algorithm known as *marching cubes*, designed by Lorensen and

Cline [12, 18] to extract surface information from 3D field of values, is then used to render the model iso-surfaces. The surface is constructed according to the following basic principle: if a point inside the desired volume has a neighboring point outside the volume, the isosurface lies between these points. This analysis is performed at the voxel level. An example of isosurface generated by US3D is given in Fig. 6. Such

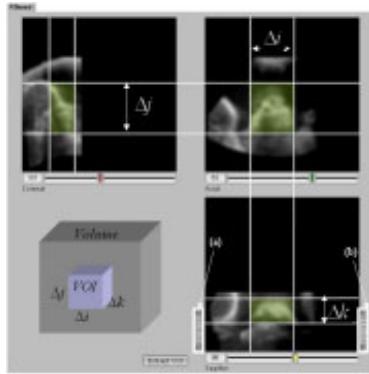


Figure 5: Selecting the volume of interest (VOI).

surface can be saved using various formats (currently VTK binary and VRML files are supported).

4 US3DTouch

The US3DTouch software has been developed to allow users to physically interact with any fetal model extracted using the US3D software. The standard proxy and god-object algorithms [17, 25] have been implemented and tested on various fetal models. Particular care has been placed on creating a stable haptic interaction. This is made difficult by the number of polygons that typically make up a fetal model, which is on the order of several tens of thousand, and by the consequent problems in creating fast ($> 1\text{KHz}$) collision detection algorithms. In order to limit such problem two different approaches have been followed:

- the number of triangles making up the system can be considerably reduced (see Fig. 8). In order to avoid cusps or other unwanted shapes due

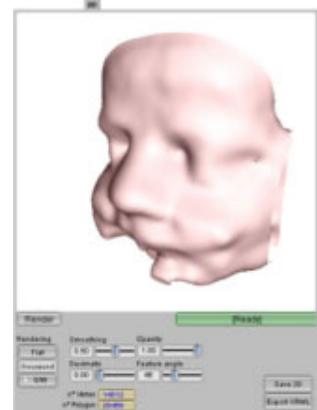


Figure 6: Isosurface extracted

to the decimation process, a smoothing procedure is used [3, 14].

- fast collision detection algorithms are used. More specifically OBB-tree [8, 9] are used to make the process faster (see Fig. 7). Note that this is made simpler by the fact that, even though the fetal model feels compliant to the user, interaction forces are computed using a static shell representing the fetus.

The system is PHANTOM based (see Fig. 1) but Delta devices [5] can be easily supported.

Various visual and haptic effects are added to the fetal model in order to make the overall simulation more realistic:

- As previously mentioned, the surface of the model is smoothed in order to eliminate bumps due to noise.
- The contact stiffness varies throughout the fetal model. This allows us to create realistic effects, such as making the fetus head feel stiffer than the rest of the body.
- A heart-rate effect is haptically simulated. More specifically the heart-rate of the fetus is directly measured and a pre-processed in order to decompose the signal in its principal components through standard FFT techniques.

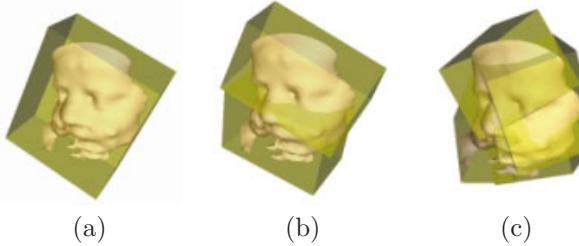


Figure 7: The Oriented Bounding Box tree. (a) Level zero. (b) Level one. (c) Level two.

Such signal is then haptically added to the standard force feedback due to contact with the fetus. While the frequency of the heart-beat signal does not change throughout the body, its amplitude is inversely proportional to the distance between fetus heart location and current contact point with the 3D model.

- The visual feedback is greatly improved by using graphical textures obtained by pictures of new born babies. Similarly, haptic textures are added to the fetal model in order to make its surface feel like human skin.

It is important to note that while the effects described above usually accomplish the purpose of making the simulation more realistic, at the current stage of the project, not all of such effects have a realistic base, i.e. properties such as varying stiffness and skin texture are not tuned according to real parameters of the fetus. For this reason the Fetouch system is not currently been used as a diagnostic tool.

5 Current limitations and future work

As previously mentioned, the current system has been created as a tool for mothers to better interact with 3D models of their fetus and not as a diagnostic tool. While the diagnostic purpose is an incredibly fascinating perspective, it is far from being a reality. Various challenges must be met in order to solve such problem. More reliable techniques to

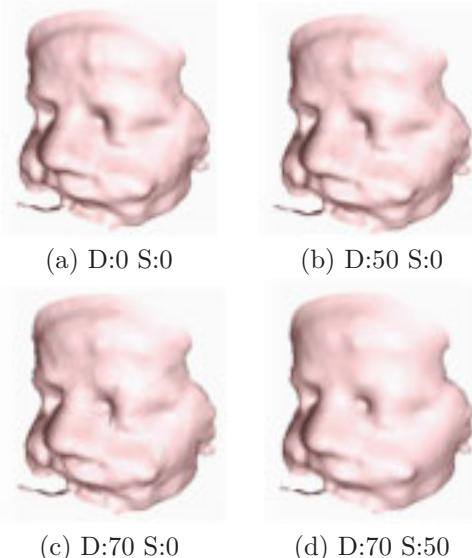


Figure 8: The number of triangles are: (a) 34167 (b) 23629 (c) 16702 (d) 3192. D is the decimation factor and S the smoothing factor.

simulate deformable objects must be developed along with procedures for in-vivo identification of stiffness parameters for the specific subject being modeled (be it a fetus or a generic human organ). Such issues will be subject of future investigation.

Acknowledgments

This research was partially supported by the Italian Ministero dell'Università e della Ricerca and by Monte dei Paschi di Siena Foundation. Authors wish to thank Filiberto Maria Severi and Felice Petraglia of the Dipartimento di Pediatria, Ostetricia e Medicina della Riproduzione of the University of Siena for their invaluable support in testing the Fetouch system.

References

- [1] R.S. Avila and L.M. Sobierajski. A haptic interaction method for volume visualization. *IEEE Proc. of Visualization 96'*, pages 197–204, 1996.

- [2] K. Baba. Basis and principles of three dimensional ultrasound. In *Three dimensional ultrasound in Obstetrics and Gynecology*, pages 1–20. Carnforth: Parthenon Publishing, 1997.
- [3] Thomas Bulow. Spherical diffusion for 3d surface smoothing. pages 163–169.
- [4] Stephane Cotin, Herve Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999.
- [5] Force Dimension. The delta system. www forcedimension com.
- [6] Wildermuth S. Bruyns C. Montgomery K. et al. Patient specific surgical simulation system for procedures in colonoscopy. In *Vision, Modeling, and Visualization*, Stuttgart, Germany, November 2001.
- [7] A.H. Gee, R.W. Prager, G.M. Treece, and L. Berman. Narrow-band volume rendering for freehand 3d ultrasound. *Computers and Graphics*, 26(3):463–476, June 2002.
- [8] S. Gottschalk, M.C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, 1996.
- [9] Arthur Gregory, Ming C. Lin, Stefan Gottschalk, and Russell Taylor. A framework for fast and accurate collision detection for haptic interaction. page 8.
- [10] Rami Hietala. Virtual laboratory. Technical Report 36, Medical Imaging Research Group, Oulu University Hospital, 1999.
- [11] Arie E. Kaufman. Voxels as a computational representation of geometry. page 45, 1997.
- [12] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(3), 1987.
- [13] National Electrical Manufacturers Association, 1300 N. 17th Street, Rosslyn, Virginia 22209 USA. *Digital Imaging and Communications in Medicine — DICOM*. URL: medical.nema.org.
- [14] Yutaka Ohtake, Alexander G. Belyaev, and Ilia A. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. page 9.
- [15] R. Prage, A. Gee, G. Treece, and L. Berman. Freehand 3d ultrasound without voxels: voulme measurement and visualization using Stradx system. *Ultrasoundics*, 40(1-8):109–115, May 2002.
- [16] R.N. Rankin, A. Fenster, D.B. Downey P.L. Munk, M.F. Levin, and A.D. Veltet. Three-dimensional sonographic reconstruction: techniques and diagnostic applications. *American Journal of Roentgenology*, 161(4):695–702, October 1993.
- [17] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. *Computer Graphics*, 31(Annual Conference Series):345–352, 1997.
- [18] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, an object-oriented approach to 3D graphics*. Prentice-Hall Inc., 1998.
- [19] H. Steiner, A. Staudach, D. Spitzer, and H. Schaffer. Three dimensional ultrasound in obstetrics and gynaecology: technique, possibilities and limitations. *Human Reproduction*, 20(9):923–936, 1994.
- [20] M. Sweet, C.P. Earls, and B. Bill Spitzak. *FLTK 2.0.0 Programming Manual (revision 11)*. Copyright 1998–2002 by Bill Spitzak et al.
- [21] Sensable Technologies. The phantom system. www.sensable.com.
- [22] Università di Siena. *The FeTOUch software*, 2002. download at www.dii.unisi.it/prattichizzo/haptic/FeTOUch.html.
- [23] Q. Wang. Translation of graphic to haptic boundary representation. Master’s thesis, McGill University, 1999.
- [24] D. Yi and V. Hayward. Skeletonization of volumetric angiograms for display. *Computer Methods in Biomechanics and Biomedical Engineering*, 2002. In press.
- [25] C. Zilles and J. Salisbury. A constraintbased god-object method for haptic display. In *Proc. IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, volume 3, pages 146–151, 1995.

Bibliografia

- [1] Nature 413. Transatlantic robot-assisted telesurgery. pages 379–380.
- [2] R.S. Avila and L.M. Sobierajski. A haptic interaction method for volume visualization. *IEEE Proc. of Visualization 96'*, pages 197–204, 1996.
- [3] K. Baba. Basis and principles of three dimensional ultrasound. In *Three dimensional ultrasound in Obstetrics and Ginecology*, pages 1–20. Carnforth: Parthenon Publishing, 1997.
- [4] Thomas Bulow. Spherical diffusion for 3d surface smoothing. pages 163 – 169.
- [5] Force Dimension. The delta system. [www.forcedimension.com](http://www forcedimension.com).
- [6] FLTK. Fast light tool kit. <http://www.fltk.org>.
- [7] A.H. Gee, R.W. Prager, G.M. Treece, and L. Berman. Narrow-band volume rendering for freehand 3d ultrasound. *Computers and Graphics*, 26(3):463–476, June 2002.
- [8] Enrico Gobbetti, Piero Pili, and Riccardo Scateni. Tecniche di visualizzazione volumetrica di carotaggi. page 8, 1997.
- [9] S. Gottschalk, M.C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96*, 1996.

- [10] Arthur Gregory, Ming C. Lin, Stefan Gottschalk, and Russell Taylor. A framework for fast and accurate collision detection for haptic interaction. page 8.
- [11] Rami Hietala. Virtual laboratory. Technical Report 36, Medical Imaging Research Group, Oulu University Hospital, 1999.
- [12] Doug L. James and Dinesh K. Pai. Pressure masks for point-like contact with elastic models. (1-4).
- [13] Thomas Jansen, Bartosz von Rymon-Lipinski, Zdzislaw Krol, Lutz Ritter, and Erwin Keeve. An extendable application framework for medical visualization and surgical planning. page 9.
- [14] Arie E. Kaufman. Voxels as a computational representation of geometry. page 45, 1997.
- [15] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Press*, pages 163 – 169, 1987.
- [16] Per Mattsson and Andreas Eriksson. Segmentation of carotid arteries from 3d and 4d ultrasound images. Master's thesis, Department of Electrical Engineering Linköpings Universitet, SE-581 83, Linköping, Sweden, 2002.
- [17] National Electrical Manufacturers Association, 1300 N. 17th Street, Rosslyn, Virginia 22209 USA. *Digital Imaging and Communications in Medicine — DICOM*. URL: medical.nema.org.
- [18] Novint Technologies. *e-Touch Sono*, 2002. URL: www.novint.com.
- [19] Yutaka Ohtake, Alexander G. Belyaev, and Ilia A. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. page 9.

- [20] Jason P.Fritz. Haptic rendering techniques for scientific visualization. Master's thesis, University of Delaware, 1996.
- [21] Piero Pili. Un algoritmo ray tracing per la visualizzazione di isosuperfici di campi scalari tridimensionali. Technical report, Scientific Visualization Group CRS4 (Centre for Advanced Studies and Research in Sardinia), Via N. Sauro, 10 Cagliari Italy.
- [22] R. Prage, A. Gee, G. Treece, and L. Berman. Freehand 3d ultrasound without voxels: volume measurement and visualization using Stradx system. *Ultrasonics*, 40(1-8):109–115, May 2002.
- [23] C.M. Pugh. E-pelvis: The case of the pelvic simulator. In *The Virtual Body: The Next Generation of Medical Imaging and Simulation*, Stanford, CA, 2001. SUMMIT, Stanford CA. Oral presentation.
- [24] R.N. Rankin, A. Fenster, D.B. Downey P.L. Munk, M.F. Levin, and A.D. Velle. Three-dimensional sonographic reconstruction: techniques and diagnostic applications. *American Journal of Roentgenology*, 161(4):695–702, October 1993.
- [25] Randy Haluck M.D. Rob Ravenscroft Ph.D. Betty Mohler Eric Crouthamel Tyson Frack Steve Terlecki Jeremy Sheaffer Roger Webster, Ph.D. Elastically deformable 3d organs for haptic surgical simulation. (1-3).
- [26] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. Haptic rendering: programming touch interaction with virtual objects. pages 123–130.
- [27] K. Salisbury and C. Zilles. A constraint-based god-object method for haptic display. page 6.

- [28] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, an object-oriented approach to 3D graphics*. Prentice-Hall Inc., 1998.
- [29] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics*. Number 646.
- [30] Siemens Medical Systems, Inc., Ultrasound Group. *SONOLINE Elega Millenium Edition*, 2000.
- [31] H. Steiner, A. Staudach, D. Spitzer, and H. Schaffer. Three dimensional ultrasound in obstetrics and gynaecology: technique, possibilities and limitations. *Human Reproduction*, 20(9):923–936, 1994.
- [32] Intuitive Surgical. Telesurgery. <http://www.intuitivesurgical.com>.
- [33] M. Sweet, C.P. Earls, and B. Bill Spitzak. *FLTK 2.0.0 Programming Manual (revision 11)*. Copyright 1998-2002 by Bill Spitzak et al.
- [34] SensAble Technolgies. *GHOST SDK 3.1Programmer's Guide Version 3.1*. Copyright 1996-2001 SensAble Technolgies.
- [35] Sensable Technologies. The phantom system. www.sensable.com.
- [36] Philippe Thevenaz and Michael Unser. High-quality isosurface rendering with exact gradient. *Swiss Federal Institute of Technology Lausanne*, page 4.
- [37] Trolltech. Qt. www.trolltech.com.
- [38] Stanford University. Collision detection. <http://www.stanford.edu/~jgao/collision-detection.html>.
- [39] Università di Siena. *The FeTOUch software*, 2002. download at www.dii.unisi.it/prattichizzo/haptic/FeTOUch.html.

- [40] P. Volino, M. Courchesne, and N. M. Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proc. SIGGRAPH'95*, pages 137–144, 1995.
- [41] VTK. Visualization toolkit. <http://www.kitware.com>.
- [42] Q. Wang. Translation of graphic to haptic boundary representation. Master's thesis, McGill University, 1999.
- [43] D. Yi and V. Hayward. Skeletonization of volumetric angiograms for display. *Computer Methods in Biomechanics and Biomedical Engineering*, 2002. In press.
- [44] Terry S. Yoo and David T. Chen. Interactive 3d medical visualization: A parallel approach to surface rendering 3d medical data. page 6, 1994.