UNIVERSITÀ DEGLI STUDI DI PISA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

TESI DI LAUREA

Progettazione di un sistema CAD per le colonscopie virtuali

Candidato
Massimo Cesaraccio

Relatori
Prof. Antonina Starita
Dott. Franco Alberto Cardillo

Controrelatore
Prof. Maria Simi

Anno Accademico 2005/2006

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'opportunità di svolgere questa tesi mi si è presentata in un momento della mia vita e carriera universitaria in cui era molto forte il bisogno di verificare l'utilità del prodotto dei miei sforzi. L'illusione di poter contribuire, seppur minimamente, al miglioramento della qualità della vita, unita alla voglia di cimentarsi in un lavoro ambizioso e impegnativo, mi ha portato ad accettare questa opportunità.

Durante tutto lo svolgimento del lavoro ho avuto modo di confrontarmi con problemi di diversa natura, non solo tecnica, molti dei quali per me del tutto nuovi. Alcuni di essi legati alla complessità dell'argomento trattato, altri alla libertà lasciatami nell'approccio al problema. Proprio quest'ultimo aspetto ha reso il lavoro ancora più stimolante e ha contribuito a un'importante crescita personale e tecnica.

La fase di sviluppo ha portato ad affrontare problemi di prestazioni, usabilità e riuso del prodotto, in un campo, l'elaborazione di immagini, per me del tutto nuovo. Sono state sviluppate soluzioni per la lettura di dataset volumetrici di tipo biomedico, per il filtraggio delle immagini, per la ricostruzione e visualizzazione tridimensionale e per l'interazione con l'utente, senza mai dimenticare i requisiti di flessibilità ed espandiblità. Si è infatti lavorato per rendere l'applicazione prodotta una base, dalla quale ripartire per sviluppare soluzioni per problemi diversi.

Il capitolo 2 introduce il problema della colonscopia virtuale dal punto di vista medico ed espone i requisiti individuati per un sistema di colonscopia

virtuale. Il capitolo 3 descrive i dataset di tipo biomedico ed in particolare i dataset nello standard DICOM. Nel capitolo 4 sono esposte e motivate le scelte progettuali e sono descritti brevemente i design pattern usati per la progettazione. Il capitolo 5 presenta gli algoritmi di image processing applicati alle immagini prima della ricostruzione 3D. Il capitolo 6 descrive la tecnica usata per la ricostruzione tridimensionale del colon. Il capitolo 7 presenta l'architettura dell'applicazione tramite diagrammi UML dei sottosistemi e una breve descrizione delle classi.

Capitolo 2

Colonscopia Virtuale

Le moderne tecnologie consentono di acquisire immagini ad alta definizione di sezioni del corpo, con una precisione tale da rendere rappresentabili in modo chiaro anche strutture molto piccole.

Per questo motivo un numero crescente di esami clinici, come la Tomografia Computerizzata (TC) e la Risonanza Magnetica (RM), è oggi basato su analisi di immagini. L'effettiva utilità di questo tipo di esami dipende dalla qualità delle immagini stesse e dall'abilità del radiologo che le interpreta. Infatti esami come TC o RM producono un elevato numero di immagini e la loro ispezione può risultare molto lunga. L'accuratezza dell'esame può quindi risultare compromessa da errori umani, dovuti a stanchezza o distrazione.

Per ridurre la probabilità di errore si ricorre solitamente a una doppia lettura di uno stesso esame: due radiologi analizzano in modo indipendente lo stesso esame, per poi confrontare i due risultati e rilevare eventuali errori o distrazoni. Un approccio alternativo consiste nell'usare il computer come un secondo radiologo. Il risultato di un'analisi computerizzata delle immagini può agevolare considerevolmente il compito del radiologo.

Sistemi progettati per fornire al radiologo un supporto di questo tipo in fase di diagnosi, vengono detti sistemi di *computer assisted diagnosis*, o *(CAD)*. Le funzionalità offerte da un sistema CAD offrono supporto al radiologo sia per quanto riguarda l'ispezione dei dati sia per quanto riguarda la diagnosi vera e propria. Per il primo dei due aspetti ci si riferisce principalmente alle tecniche che permettono un'osservazione più rapida ed efficace

dei volumi che costituiscono gli esami, fornendo diversi tipi di visualizzazione tra loro integrati. Per quanto riguarda il secondo aspetto si può distinguere tra i sistemi che mirano a fornire due tipi di supporto:

Automated diagnosis consiste nell'effettuare la diagnosi in modo del tutto automatico, e fornire risultati che devono essere considerati attendibili. Mira a sostituire completamente il radiologo nella fase di diagnosi;

Assisted diagnosis consiste nel fornire un risultato che costituisca un suggerimento per il radiologo, in modo che questi possa intervenire per validare la diagnosi. Mira essenzialmente a ridurre lo spazio di ricerca, in modo da semplificare e velocizzare l'ispezione dei dati.

L'obiettivo del lavoro oggetto di questa tesi è stato inizialmente definito come lo sviluppo di un sistema di CAD per la colonscopia virtuale. Nel corso del capitolo verrà illustrato come i requisiti per un sistema di questo tipo hanno modificato, parzialmente, gli obiettivi e lo svolgimento del lavoro.

L'esame di colonscopia tradizionale viene detto colonscopia ottica, in quanto il colon del paziente è ispezionato dall'interno mediante l'uso di una telecamera di piccole dimensioni.

Un sistema di colonscopia virtuale permette invece una visita virtuale del colon, affiancando alla visualizzazione delle immagini 2D acquisite attraverso un esame TC, una rappresentazione tridimensionale del colon stesso. In esami di questo tipo il radiologo può ispezionare il colon posizionando il punto di vista dell'osservatore all'interno del lume.

Attraverso la colonscopia virtuale è possibile ispezionare zone del colon inaccessibli durante una colonscopia tradizionale. Essendo meno invasivo, può essere accettato più facilmente da persone che, avendo un alto rischio cancro al colon, devono essere sottoposte ad un monitoraggio continuo (screening).

2.1 Il cancro al colon

Come riportato in [TI] il tumore, o *neoplasia*, consiste in una crescita anomala, incontrollata e progressiva di un gruppo di cellule.

Il processo di continuo e disordinato accrescimento è un elemento comune a tutti i tipi di tumori ed è la principale differenza tra cellule sane e cellule tumorali.

Quando le cellule tumorali presentano la capacità di separarsi dal tumore originale e di diffondersi nel corpo attraverso il sangue o il sistema linfatico si parla di tumore maligno o *cancro*. Le cellule tumorali di questo tipo possono raggiungere zone lontane da quella di origine nelle quali riprodursi e formare nuovi tumori (*metastasi*).

2.1.1 Anatomia del colon

Lo scopo principale del sistema digerente è di estrarre i nutrienti dal cibo ingerito e riversarli nel flusso sanguigno in modo che possano essere distribuiti alle cellule nell'organismo.

Il colon fa parte dell'intestino crasso ed è composto da quattro sezioni: colon ascendente, colon trasverso, colon discendente e colon pelvico o sigma (vedi figura 2.1).

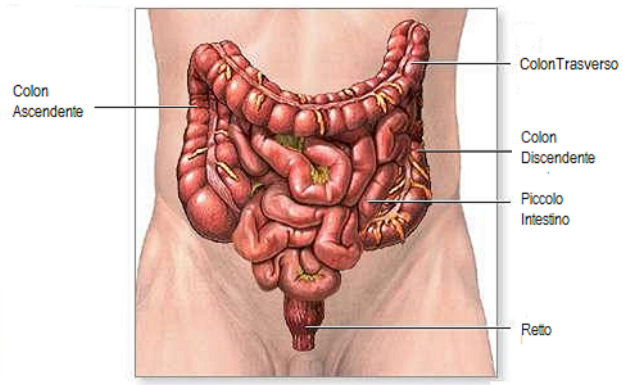


Figura 2.1: Il Colon

Il colon ascendente comincia dove finisce il cieco, all'altezza della valvola ileocecale, risale fino alla flessura destra diventando trasverso. Il trasverso prosegue da destra verso sinistra fino alla flessura sinistra, dove comincia il colon discendente. Questo prosegue verso il basso diventando sigma prima di terminare nel retto.

Osservando la figura 2.1 si intuisce come il colon sia caratterizzato da una struttura complessa, ricca di curve. In molti casi la struttura può rivelarsi particolarmente contorta, con tratti del colon che si sovrappongono l'uno all'altro e che si restringono, crendo talvolta delle occlusioni.

Le continue curve del colon e le restrizioni che si incontrano lungo il suo perrcorso, non consentono un'eplorazione tradizionale nei tratti iniziali. Questa caratteristica, inoltre, rende molto difficile anche la navigazione virtuale: spesso in una regione di curvatura il percorso del lume è riconoscibile solo in un piano perpendicolare a quello di acquisizione. In questi casi capita che, osservando immagini consecutive, le sezioni del tratto di colon coinvolto si riducano progressivamente di diametro fino a scomparire. Il tratto che segue la curva sarà infatti contenuto nelle stesse immagini del tratto precedente la curva, probabilmente affiancato ad esso (vedi fig. 2.2, pag. 14).

2.1.2 Tumore al colon

Il tumore al colon può avere origine in una qualsiasi delle quattro sezioni che lo compongono. Le pareti del colon, così come di tutto l'intestino crasso, sono costituite da strati concentrici. I tumori derivano dalla trasformazione delle cellule epiteriali del più interno di questi strati, detto tonaca mucosa, e vengono detti adenoma o adenocarcinoma a seconda che siano di forma benigna o maligna. Il 95% dei casi di tumore al colon sono adenocarcinomi [ACS].

La maggioranza dei tumori maligni del colon-retto insorge a partire da lesioni, spesso inizialmente del tutto benigne, dette polipi adenomatosi (fig. 2.3, pag. 15).

Si tratta di piccoli agglomerati di tessuto che proliferano all'interno del canale intestinale. Il fatto che i polipi adenomatosi siano spesso i precursori del tumore ha una notevole importanza pratica poiché l'asportazione endoscopica dei polipi è in grado di prevenire lo sviluppo di un tumore maligno. Poiché l'individuazione e la rimozione tempestiva di polipi è determinante nel prevenire tumori, risulta evidente l'importanza che assume un programma di screening su pazienti considerati a rischio. Purtroppo, a causa dei tempi necessari per effettuare ed analizzare una colonscopia virtuale, il controllo

degli esami di un gran numero di pazienti è un'operazione che richiede molto tempo. Questo fa sì che i lunghi tempi di risposta per un esame influenzino negativamente le possiblità di una diagnosi precoce.

Tuttavia un programma di screening che sia in grado di monitorare tutte le persone a rischio è molto importante. Infatti il tumore al colon è comune in tutto il mondo, con incidenza variabile: è comune in occidente e più raro in Asia e Africa. A livello mondiale i tumori di questo tipo risultano quarti come incidenza tra tutti i tipi di tumore. Tra i tumori rappresentano la seconda causa di morte negli uomini dopo il cancro al polmone e la terza nelle donne dopo il cancro al seno e al polmone [ACS].

Ogni anno nel nostro paese circa 27.000 persone si ammalano di tumore del colon-retto e, di queste, più della metà muore a causa della malattia. Si valuta che ogni anno vi siano circa 45 nuovi casi ogni 100.000 abitanti, con una maggiore incidenza nell'Italia centrosettentrionale. La tabella 2.1 mostra l'incidenza dei tumori al colon in Italia per fascia d'età.

Età	casi ogni 100000 abitanti
< 45	2
45-54	20
55-64	55
65-74	120
> 74	200

Tabella 2.1: Incidenza dei tumori al colon in Italia

La malattia colpisce con maggiore frequenza individui di sesso maschile.[TI]

Fattori di rischio noti sono l'età, una dieta troppo ricca di grassi di origine animale, mancanza di esercizio fisico, obesità, fumo e alcool, nonchè fattori genetici ereditari. In particolare persone con una storia familiare di tumori al colon e di età superiore ai 50 anni dovrebbero ricorrere a controlli annuali in grado di effettuare la diagnosi precoce di eventuali tumori.

2.2 La colonscopia vitruale

La colonscopia virtuale è preceduta da una fase di preparazione: il colon del paziente viene gonfiato con aria, in modo che le pareti risultino il più distese possibile. Nelle immagini il colon è riconoscibile come regioni di colore nero all'interno dell'addome. Il nero corrisponde infatti al colore su cui viene mappato il segnale di risposta dell'aria. Se il colon non si gonfia in modo omogeneo, alcuni tratti possono collassare, rendendo impossibile l'individuazione del lume nelle immagini. Inoltre eventuali residui presenti nel colon possono essere scambiati per polipi, in quanto caratterizzati nelle immagini dagli stessi toni di grigio. Per ridurre il numero di fasi positivi si ricorre, talvolta, all'uso di un mezzo di contrasto che modifichi il valore di risposta dei residui, Questa procedura non viene di norma usata per lo screening, a causa dei costi e della dilatazione dei tempi di preparazione se applicata a un numero elevato di pazienti.

2.3 Analisi del problema

Il lavoro di tesi è stato sviluppato in collaborazione con il Dipartimento di Radiologia Diagnostica ed Interventistica dell'Università di Pisa. Le interviste con i radiologi di tale dipartimento ha permesso di evidenziare le caratteristiche fondamentali di un sistema CAD per la colonscopia virtuale. Dalle informazioni emerse dalle interviste e dallo studio dello stato dell'arte [YN01, YNM+01, PBR+04, GTA+, GTA+01] è stato individuato l'elenco delle caratteristiche che il sistema deve offrire:

- leggere i dati prodotti dai dispositivi di acquisizione;
- visualizzare contemporaneamente le immagini e la rappresentazione tridimensionale del colon, consentendo al radiologo di muoversi all'interno del modello 3D;
- consentire al medico di interagire con i dati mediante entrambe le rappresentazioni, sfruttando le informazioni combinate che esse forniscono;

- consentire al medico di selezionare una *Region of Interest* (ROI), ossia una regione del volume di particolare interesse, sia in 2D che in 3D;
- consentire l'analisi della ROI selezionata.

Tutte le caratteristiche elencate, ad eccezione di quelle che riguardano direttamente la colonscopia virtuale, sono in realtà di carattere generale, sono cioè applicabili a tutti i sistemi CAD dedicati all'elaborazione di dati volumetrici.

Si è quindi deciso di ridurre gli obiettivi e di sviluppare uno strumento che potesse essere applicato, in modo più generale, alla visualizzazione e alla manipolazione di dataset volumetrici di tipo biomedico. Ciò ha modificato gli obiettivi iniziali e ha portato alla progettazione ed implementazione di un framework generale, per l'ispezione e la manipolazione di questo tipo di dati. Il sistema è stato inoltre specializzato nei moduli che elaborano e ricostruiscono il volume.

Questo approccio ha influenzato le modalità di progettazione dell'architettura software. I componenti dell'applicazione sono stati progettati per essere applicati a problemi differenti, e possono essere così raggruppati:

- componenti per applicazioni interattive;
- componenti per applicazioni dedicate all'elaborazione di dataset volumetrici;
- componenti specifici per sistemi CAD;
- componenti per applicazioni di colonscopia virtuale.

2.4 Caratteristiche e requisiti

Le caratteristiche del sistema possono essere suddivise nei seguenti gruppi di funzionalità:

- lettura delle informazioni dai file prodotti dai dispositivi di acquisizione;
- presentazione dei dati;

- interazione con l'utente nell'ispezione delle immagini e del volume;
- elaborazione dei dati;
- analisi di porzioni del volume.

Come già accennato i requisiti fondamentali per un sistema CAD sono la flessibilità e l'espandibilità. Questi requisiti hanno influenzato profondamente la progettazione delle soluzioni per ciascun punto.

Analizziamo quindi ciascun gruppo di funzionalità individuato:

Lettura dei dati

I dati sono memorizzati su disco secondo alcuni formati standard, in una modalità che riflette l'organizzazione dei dataset. Tali formati non sono fissi ma si evolvono nel tempo, seguendo l'evoluzione della metodologia di acquisizione negli esami clinici.

Il sistema deve perciò tenere in considerazione il fatto che i componenti che costituiscono i dataset possano mutare, o che il numero di tali componenti possa aumentare. Il supporto ai formati deve essere facilmente modificabile ed estendibile, in modo che sia possibile mantenere il sistema costantemente aggiornato rispetto alle caratteristiche degli esami.

Data la complessità dei dati, l'accesso agli stessi dovrebbe essere il più possibile semplificato, fornendo un'astrazione delle entità di informazione che costituiscono il dataset.

I formati più diffusi per la gestione di immagini mediche sono DICOM (si veda appendice [ACR00]) e Analyze.

Presentazione dei dati

I dati letti da disco devono essere visualizzati in modo chiaro ed efficace. In particolare l'ispezione dei dati volumetrici deve essere resa intuitiva.

La visualizzazione delle immagini che costituiscono i volumi deve essere adattabile a una serie di parametri, funzione del particolare dataset, a scelta dall'utente.

Le immagini ceh costituiscono i volumi acquisiti dai dataset non sono però gli unici dati visualizzabili. I dati possono essere infatti soggetti a vari tipi di elaborazioni, alcune delle quali potrebbero produrne di nuovi. Questi potrebbero essere strutturalmente simili a quelli originali o completamente diversi, costituendo magari dell'informazione integrativa o una rappresentazione differente, che andrebbe ad affiancare quella originale.

Per quanto riguarda i dati frutto di elaborazioni, le metodologie di visualizzazione non risulterebbero legate ai dettami del formato in modo così stretto come i dati originali. Ci sarebbe quindi del margine di interpretazione e di inventiva, per produrre delle visualizzazioni che offrano nuovi punti di osservazione sui dati.

In ogni caso, sia per motivi legati ai parametri imposti dal formato, sia per motivi legati alla scelta di diverse modalità di presentazione, tutte le procedure di visualizzazione devono essere estendibili e specializzabili.

Elaborazione dei dati

L'elaborazione dovrà avvenire sia tramite algoritmi generali, validi indipendentemente dal tipo di dati, sia tramite algoritmi dedicati. Questi ultimi potrebbero essere specializzati per certi tipi di esami o per tipi di visualizzazione diversi.

Pertanto l'insieme degli algoritmi di elaborazione deve essere estendibile e specializzabile a seconda delle caratteristiche dei dati che si vogliono trattare e dei risultato che si intende ottenere.

Interazione con l'utente

La complessità dei dati che un sistema CAD gestisce comporta che le operazioni che il sistema potrebbe compiere su di essi possano essere estremamente varie ed articolate. Infatti, come gli altri aspetti finora descritti, anche l'interazione tra l'utente e i dati può variare sensibilmente a seconda delle finalità e delle caratteristiche degli esami. Inoltre, poiché l'interazione risulta fondamentale per accedere alle funzionalità del sistema, si intuisce che estensioni o modifiche di altre componenti potrebbero rendere necessaria l'aggiunta di nuovi meccanismi di interazione.

Le principali funzionalità che il sistema deve offrire sono:

- possibilità di selezionare interattivamente i dataset volumetrici da trattare, a seconda dei parametri che caratterizzano l'organizzazione dei dati; per esempio deve permettere di selezionare una sorgente, come un database o un percorso su disco, da cui leggere il dataset, e da questo l'esame da visualizzare, selezionando lo studio e la serie che lo individuano;

- possibilità di modificare interattivamente la modalità di presentazione dei dati, essenziale per evidenziare aspetti peculiari degli elementi del volume o di strutture ausiliarie; per esempio, deve permettere di modificare la scala dei grigi con cui presentare le immagini su schermo o di posizionarsi interattivamente nel modello tridimensionale per osservare da vicino determinate strutture;

- possibilità di selezionare parti del volume per manipolarle in modi diversi; per esempio deve permettere di selezionare una ROI di un'immagine 2D o di un volume 3D;

- possibilità di richiamare algoritmi di elaborazione e di applicarli ai dati, a un sottoinsieme selezionato o a strutture ausiliarie prodotte dal sistema; per esempio deve permettere di estrarre caratteristiche morfologiche da una ROI di un'immagine o di una porzione di volume selezionata;

- possibilità di far circolare messaggi tra i componenti del sistema, in modo da propagare gli effetti di modifiche ed elaborazioni operate dall'utente in modo interattivo; ad esempio il risultato di un'elaborazione, come quella descritta al punto precedente, in un'altra finestra rispetto a quella attraverso la quale è stata richiesta l'elaborazione.

È evidente che le funzionalità elencate sono comuni a più esami radiologici. È quindi preferibile, piuttosto che creare un sistema ad hoc, progettare un sistema generale, da specializzare su esami radiologici di diverso tipo.

I dataset che abbiamo avuto disposizione per il lavoro di tesi sono stati prodotti da esami di tipo TC, che costituiscono i dati su cui opera la colonscopia virtuale. Il sistema è stato sviluppato basandosi sulle caratteristiche

dei dati disponibili, memorizzati secondo lo standard più diffuso: DICOM (vedi 3.3).

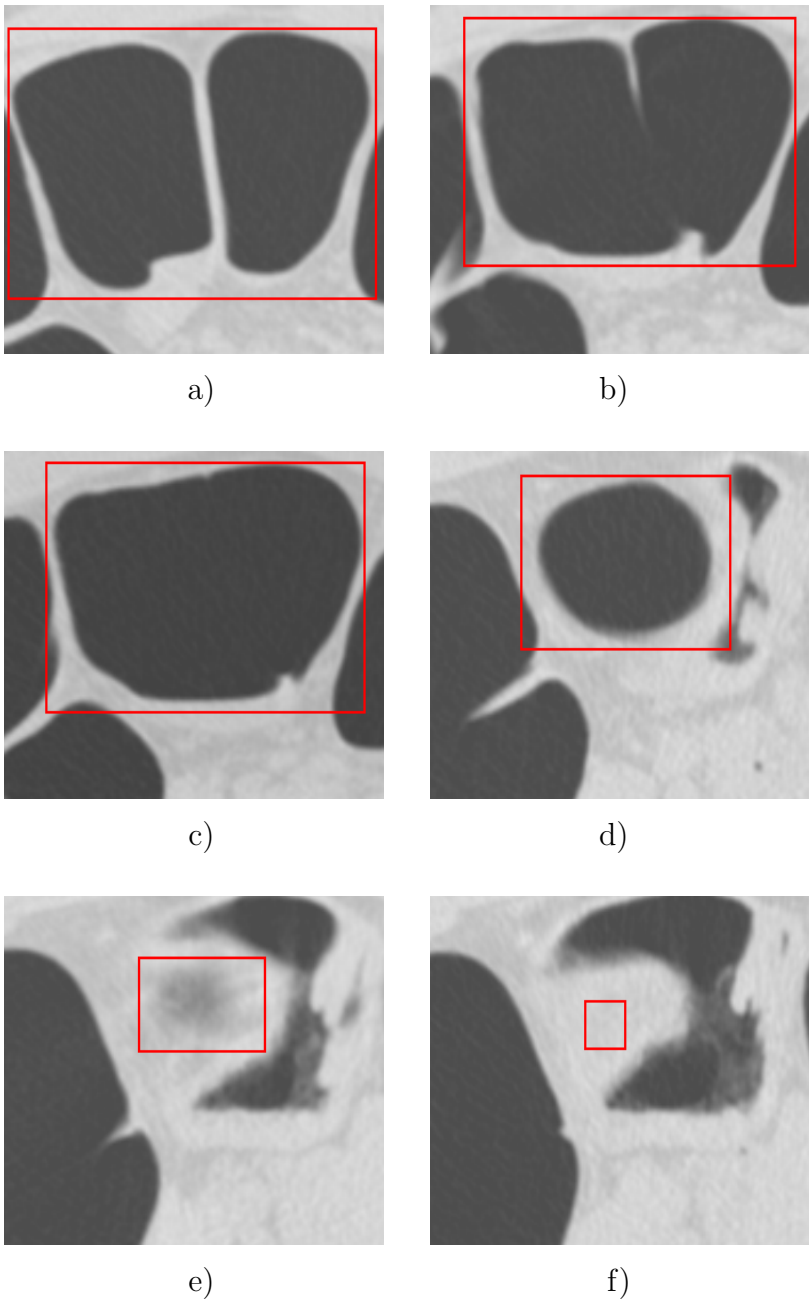


Figura 2.2: a) le due regioni al centro della figura rappresentano due tratti paralleli uniti da una curva; b) all'inizio della curva le regioni cominciano a fondersi; c) nella curva sono un unica regione; d) il diametro della regione diminuisce; e) la parete esterna della tratto curvo è quasi scomparsa; f) l'immagine successiva non contiene più sezioni del tratto curvo

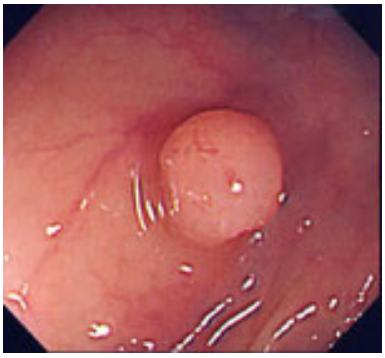


Figura 2.3: Esempio di polipo.

Capitolo 3

Dataset Volumetrici

I dati trattati dal sistema sono generati da computer, tipicamente delle workstation dedicate, che lavorano in simbiosi con i dispositivi di acquisizione delle immagini. Tali dispositivi possono essere di vario tipo e comprendono scanner per la tomografia computerizzata (TC) o per la risonanza magnetica (RM).

I dispositivi di acquisizione effettuano delle misure su una parte del corpo del paziente. Il risultato viene elaborato dalle workstation, le quali effettuano il campionamento del segnale e producono come risultato una serie di immagini digitali, sezioni del volume originale occupato dal paziente.

Nelle sezioni che seguono sarà brevemente descritta l'organizzazione di un dataset e saranno elencate le informazioni che, tra quelle tipicamente contenute nei dati, si sono rivelate più significative per lo svolgimento del lavoro.

3.1 Organizzazione del dataset

Ciascun esame di un paziente è suddiviso in studi, ciascuno dei quali può essere composto da più serie.

Le immagini appartenenti ad una stessa serie costituiscono un volume. Allo stesso modo le serie che sono generate nella stessa sessione costituiscono uno studio. Al contrario di una radiografia, che contiene una rappresentazione proiettiva bidimensionale di una parte del corpo, gli esami volumetrici

contengono la rappresentazione di un volume tridimensionale, a discapito della risoluzione di ciascuna immagine che lo compone. Per questo motivo ci si riferisce a questo tipo di dati come a dataset volumetrici.

Le informazioni di un dataset sono distribuite in un insieme di file, ciascuno dei quali contiene tutte le informazioni necessarie all'organizzazione e all'aggregazione dei dati. Come tali informazioni sono registrate nei file dipende dal formato con il quale il computer che genera i dati li memorizza. Nel corso del lavoro è stato possibile lavorare su file nel formato DICOM, che rappresenta lo standard de facto nella gestione degli esami per immagini in ambito ospedaliero. Pertanto nei successivi capitoli questo sarà l'unico formato considerato tra tutti quelli esistenti. Per una descrizione del formato DICOM si veda l'appendice A.

Indipendentemente dal formato con cui i dati sono registrati, ogni file deve comunque contenere le informazioni necessarie all'ordinamento dei dati, al fine di poter identificare ed aggregare il blocco di informazioni relativo ad un paziente e all'interno di esso gli studi, le serie e le immagini che lo compongono.

L'informazione può essere distribuita su diverse centinaia di file ed arrivare ad occupare uno spazio considerevole. In ogni file si trovano, oltre ai dati che costituiscono il volume, una grande quantità di altre informazioni, molte delle quali non risultano utili né per l'organizzazione né per la decodifica del volume. Tra queste riportiamo, come esempio, le caratteristiche della macchina che ha eseguito l'acquisizione, i dati del radiologo che ha effettuato l'esame o dati accessori riguardanti il paziente, come ad esempio il sesso.

Le informazioni più importanti riguardano:

Ordinamento come già accennato comprendono identificativi di paziente, studio e serie, più attributi che permettono di collocare sia spazialmente che temporalmente un file, e quindi un immagine, all'interno di un volume;

Decodifica comprendono le informazioni che permettono di estrarre dal file e di decodificare i dati volumetrici, ossia il valore dei *voxel* che costituiscono il volume.

Nella prossima sezione i concetti di volume e voxel saranno discussi in maniera più approfondita.

3.2 Dati volumetrici

Un volume è suddiviso da una griglia tridimensionale la cui unità di informazione è detta *voxel*. Un voxel rappresenta in un volume quello che un pixel rappresenta in un'immagine e contiene la rappresentazione discreta della misurazione effettuata nella porzione corrispondente del volume originale.

Il volume viene suddiviso in sezioni, tutte dello stesso spessore, ciascuna delle quali è memorizzata su disco come un'immagine. Il file che contiene l'immagine porta con se dell'informazione riguardante lo spessore della sezione di volume che l'immagine rappresenta. Ogni pixel dell'immagine sarà a sua volta caratterizzato da alcuni attributi, dei quali si parlerà in dettaglio nel capitolo 2.

3.3 Dataset DICOM

I file in formato DICOM contengono le informazioni caratteristiche per dataset di tipo biomedico, delle quali si è brevemente parlato nelle sezioni precedenti del capitolo. Per una descrizione del modello dei dati dello standard DICOM si rimanda all'appendice A. In questa sezione verranno descritti gli attributi più significativi e utili per la realizzazione delle seguenti fasi:

- ordinamento e raggruppamento dell'informazione;
- estrazione dai file e decodifica dei dati volumetrici;
- presentazione dei dati.

Le fasi elencate nascono dal raggruppamento di oggetti informativi dello standard e rispecchiano le fasi che l'applicazione attraversa per consentire la lettura e la presentazione di un dataset.

3.3.1 Ordinamento

Per identificare un paziente sono stati usati gli attributi elencati di seguito:

Patient ID identificativo del paziente;

Patient Name nome del paziente.

Per identificare uno studio sono stati usati gli attributi:

Study Instance UID identificativo unico dello studio;

Study Date data nella quale è stato acquisito lo studio;

Study Time ora alla quale è stato acquisito lo studio.

Per identificare una serie sono stati usati gli attributi:

Series Instance UID identificativo unico della serie;

Series Date data nella quale è stata acquisita la serie;

Series Time ora alla quale è stata acquisita la serie.

Infine per identificare un'immagine sono stati usati gli attributi:

Slice Location posizione relativa dell'immagine lungo l'asse di acquisizione, espressa in millimetri.

Instance Number numero progressivo che identifica il numero di immagine nella serie

Gli attributi appena elencati permettono di ordinare i file e le immagini in essi contenute, in modo che sia possibile ricostituire i volumi tridimensionali. Permettono inoltre di associare ogni volume al paziente e allo studio di origine.

3.3.2 Decodifica

Gli attributi del modulo Image Pixel descrivono come i pixel dell'immagine sono registrati su file. Permettono quindi di estrarre l'immagine e sono così definiti:

Rows e Columns indica il numero di righe e di colonne dell'immagine;

Bits Allocated indica numero di bit allocati per ciascun pixel dell'immagine;

Bits Stored indica il numero di bit effettivamente usati per registrare il valore di un pixel;

High Bit indica il bit più significativo tra quelli effettivamente usati;

Pixel Representation indica come viene rappresentato ciascun pixel; può assumere i valori *unsigned integer* o *complemento a due*;

Pixel Data indica l'inizio dello stream di dati che costituisce l'immagine;

Smallest Image Pixel Value indica il valore minimo tra i pixel dell'immagine;

Largest Image Pixel Value indica il valore massimo tra i pixel dell'immagine;

Nel modulo Image Plane sono definiti gli attributi che definiscono il rapporto tra un immagine e la corrispondente sezione di volume. Tali attributi permettono quindi di ricostruire il volume dalle immagini. Di seguito sono elencati e brevemente descritti :

Pixel Spacing specifica la distanza tra il centro di due pixel adiacenti secondo una relazione di vicinato, lungo le direzioni (1,0) e (0,1) del piano di acquisizione;

Pixel Aspect Ratio definisce il rapporto tra la dimensione dei lati del pixel;

Slice Thickness definisce lo spessore della sezione di volume rappresentata dall'immagine, espresso in millimetri.

Pixel Spacing, Pixel Aspect Ratio e Slice Thickness descrivono le caratteristiche dimensionali delle sezioni del volume e permettono quindi di mettere in relazione i pixel delle immagini con i voxel del volume. Sono fondamentali affinché la ricostruzione tridimensionale del volume sia fedele all'originale.

3.3.3 Presentazione

Gli attributi che definiscono come le immagini vanno presentate sono contenuti in due Information Entity : *Modality LUT* e *VOI LUT*.

Gli attributi di Modality LUT definiscono una prima fase di elaborazione: i voxel vanno trasformati dall'unità di misura dell'apparecchiatura di acquisizione a un'unità di misura standard, indipendente dall'apparecchiatura. Questa unità di misura è chiamata *Hounsfield Unit*.

Modality LUT può essere contenuta all'interno dell'information entity Image o in un information entity separato. È caratterizzata dagli attributi *Rescale Intercept* e *Rescale Slope*. I voxel originali SV (stored value) vengono trasformati in Hounsfield Units HU secondo la formula

$$\mathsf{HU} = \mathfrak{m} * \mathsf{SV} + \mathfrak{b}$$

dove $\mathfrak{m}$ e $\mathfrak{b}$ rappresentano rispettivamente Rescale Slope e Rescale Intercept.

Il segnale di origine è campionato con un numero di bit molto alto che non consente un'immediata visualizzazione sul monitor. Perciò i valori in Hounsfield Units devono poter essere elaborati in modo da poter essere stampati o visualizzati, Gli attributi fondamentali di VOI LUT sono *Window Center*, anche noto come *Window Level* e *Window Width*.

Window Center e Window Width specificano rispettivamente il centro e l'ampiezza della finestra di valori grigio che si vogliono visualizzare su schermo. Poiché infatti gli schermi tradizionali offrono solo 8 bit per la visualizzazione dei grigi, mentre i pixel letti sono tipicamente di 12 o 16 bit, non tutti i valori possono essere visualizzati contemporaneamente. Gli attributi Window Center e Window Width permettono di definire quali valori si vogliono visualizzare, trasformandoli sulla scala di grigi che un classico schermo di computer è in grado di visualizzare.

Capitolo 4

Progettazione del sistema

La metodologia usata per la progettazione del sistema è stata dettata dalle esigenze di flessibilità ed espandibilità che questo tipo di applicazioni richiede.

Nel corso del capitolo verà illustrata la strada scelta per modellare l'applicazione in modo da soddisfare i requisiti descritti nella sezione 2.4.

4.1 Scelte progettuali

Un sistema di CAD è un'applicazione complessa, caratterizzata da funzionalità diverse ma tra loro correlate, che di conseguenza copre molteplici problematiche progettuali.

La complessità delle funzionalità che emergono dai requisiti, unita alle caratteristiche di flessibilità ed espandibilità richieste, ha portato ad attraversare una lunga fase di progettazione che facilitasse lo sviluppo dell'applicazione.

Dai gruppi di funzionalità individuati e descritti nella sezione 2.4, sono stati definiti i moduli in cui l'applicazione è divisa. In questo capitolo ci limiteremo ad illustrare come sono state affrontate le problematiche sollevate da ciascun gruppo, rimandando la descrizione dei moduli al capitolo 7.

Nella fase di progettazione si è fatto ricorso a *design pattern* noti, che hanno fortemente ispirato il modo in cui i dati, gli attori del sistema e le comunicazioni tra di essi sono stati modellati. Nella sezione 4.2 verranno

descritti i pattern usati, specificanfo a quali problemi sono stati applicati e in che modo.

Quando nel seguito di questo capitolo e nei capitoli successivi si incontreranno termini come *ereditarietà*, *polimorfismo* e *overloading*, sarà implicito che ci si riferisce a termini noti della programmazione orientata agli oggetti. Il linguaggio di programmazione scelto è il C++. Le caratteristiche del linguaggio che hanno portato a tale scelta sono:

- l'essere un linguaggio orientato agli oggetti;
- equilibrio tra performance e flessiblità;
- gestione esplicita della memoria, fondamentale per via delle dimensioni dei dati trattati;
- integrazion naturale con la libreria OpenGL;
- l'esistenza di un gran numero di librerie esterne che estendono le funzionalità del linguaggio.

In particolare la velocità di elaborazione ha costituito, fin dall'inizio, un requisito fondamentale tra quelli emersi dalle inerviste con i radiologi. Le librerie a cui ci si è apoggiati per lo sviluppo sono brevemente introdotte nell'appendice B.

Lettura dei dati

Il sistema di lettura delle informazioni è composto da diverse classi. Il proliferare di classi ha rappresentato un problema dal punto di vista progettuale. Infatti, identificato l'insieme delle classi come un modulo indipendente dal resto dell'applicazione, ci si è accorti che l'accesso al modulo risultava poco intuitivo. Per uno sviluppatore che avesse voluto sfruttare le funzionalità del modulo, sarebbe risultato difficile capire quali classi ne costituissero l'interfaccia e quali invece fossero parte dei suoi meccanismi interni.

Per rendere chiaro ed intuitivo l'uso del modulo e l'accesso alle informazioni lette, è opportuno fornire un unico punto di acceso al modulo, che permetta di accedere in modo indiretto alle informazioni.

Presentazione dei dati

Le funzionalità appartenenti a questa categoria hanno dato origine a due moduli distinti:

- un modulo che fornisce funzionalità di utilità generale per applicazioni interattive, come ad esempio la gestione del'input del mouse, lo splitting delle finestre di visualizzazione e la propagazione di eventi attraverso l'albero delle finestre;
- un modulo che fornisce strumenti specifici per la visualizzazione di contenuti per applicazioni CAD, come, ad esempio, visualizzazione di immagini e di rappresentazioni tridimesionali degli elementi;

Entrambi i moduli dovranno fornire caratteristiche di flessibilità per adattarsi a modalità di visualizzazioni diverse.

Un modo classico per ottenere questo risultato fa ricorso al subclassing. Fornendo una classe base per la visualizzazoine, le classi specifiche per determinate modalità costituirebbero estensioni di tale classe. Lo svantaggio dei meccanismi di ereditarietà è rappresentato dal fatto che, per ottenere piccole modifiche delle modalità di visualizzazione esistenti, è necessario estendere le classi che le implementano. Questo comporta la creazione di un albero di derivazione che può rendere difficoltosa la scelta di una classe da cui estendere, in particolar modo se si vogliono combinare le caratteristiche di due classi appartenenti a due rami diversi dell'albero di derivazione.

In questo caso la soluzione tipica può seguire due approcci:

- usare l'ereditarietà multipla, che permette di definire delle classi come estensione di più classi base, anche dello stesso albero di derivazione. Questo approccio non è sempre d'aiuto, talvolta può invece portare altri problemi. Infatti se per esempio si eredita da due classi, foglie di due rami diversi dell'albero, queste potrebbero aver definito dei membri con funzionalità simili o addirittura con gli stessi nomi. In questo caso sono le specifiche del linguaggio di programmazione usato a determinare il risultato e lo sviluppatore perde parte del controllo sul comportamento. Inoltre per questo tipo di derivazioni si deve spesso ricorrere a ereditarietà privata. Infatti derivare da una classe in modo pubblico

comporta definire il tipo della sottoclasse come una specializzazione del tipo della classe base o delle classi base. Derivando la classe A da B e C, si modella il fatto che A *è* una B e una C, e questo non è sempre quello che si vuole ottenere;

- definire le classi per composizione, che limita l'uso di subclassing e permette di definire una classe come composta da altre classi, di tipo diverso. Una classe composta implementa le proprie funzionalità in base alle implementazioni delle classi che la compongono. In questo modo variando i componenti si puo' variare il comportamento della classe. Può anche trarre vantaggio dalla tecnica nota come *pimpl*, o pointer to implementation, che permette di manterere separate definizione e implementazione di una classe [Mey05].

L'approccio scelto è il secondo, in quanto permette di contenere il numero delle classi e consente un maggior riuso del codice. Applicato alla visualizzazione, la sostituzione dei componenti della classe che la gestisce, modifica le procedure di visualizzazione.

Interazione con l'utente

Le funzionalità appartenenti a questa area sono strettamente legate a quelle inerenti la visualizzazione. Per questo motivo alcune delle funzionalità sono divenute parte dei moduli prodotti dall'analisi dell'area suddetta. In particolare l'interazione che permette di modificare interattivamente i parametri di visualizzazione non poteva essere scissa in un modulo a sé stante.

Anche il modo in cui i componenti dell'interfaccia dell'applicazione gestiscono l'input dell'utente deve essere configurabile ed estendibile. Infatti le operazioni da svolgere possono dipendere dal contesto in cui opera l'applicazione, dai dati che si manipolano e visualizzano o semplicemente dal tipo di implementazione che chi sviluppa l'applicazione vuole fornire.

Come per la visualizzazione, anche questo aspetto trarrebbe vantaggio dalla modellazione del sistema di input per composizione. In questo caso, comporre la reazione all'input come un insieme di classi, permetterebbe di attivare e disattivare i componenti, rendendo il sistema dipendente dal

contesto e dallo stato in cui l'applicazione si trova. Inoltre l'incapsulare comportamenti di reazione in oggetti indipendenti permette ad altri elementi dell'applicazione di fornire moduli di input, verso i quali potersi registrare come listener.

Inoltre un sistema di questo tipo deve permettere a piú elementi del sistema di accedere ai dati per leggerli e modificarli. Pertanto la modifica va notificata a tutti gli elementi coinvolti. Per esempio gli elementi di visualizzazione potrebbero richiedere di essere sempre aggiornati in tempo reale di avvenute modifiche sui dati.

Una circolazione capillare dell'informazione comporta difficoltà di controllo. Inoltre, legare gli elementi del sistema alle politiche di comunicazione renderebbe difficoltoso il riuso di tali elementi in caso di cambiamento delle politiche stesse. Introdurre un punto di centralizzazione per le comunicazioni consentirebbe di incorporare le suddette politiche in un oggetto, con il compito di instradare l'informazione verso altre parti del sistema.

4.2 Design Patterns

In [GHJV05] si definisce così il concetto di design pattern:

> Un design pattern attribuisce un nome ad un problema di progettazione, astrae e identifica gli aspetti principali di una struttura progettuale utile per la creazione di un progetto ad oggetti riusabile.

Ogni design pattern identifica le classi che vi partecipano, le loro istanze, i loro ruoli, le collaborazioni e le distribuzione delle responsabilità.

Descriveremo ora brevemente gli aspetti più significativi dei pattern che si sono rivelati utili alla progettazione del sistema, fornendo per ciascuno di essi un esempio di applicazione.

4.2.1 Strategy

Il pattern Strategy permette di definire una famiglia di algoritmi, incapsularli e renderli intercambiabili e indipendenti dal client che li usa, in modo trasparente.

Permette cioè di modellare diverse entità, che differiscono solo nel comportamento, come un'unica classe, il cui comportamento cambia a seconda delle classi Strategy che la compongono. Una classe così costituita viene assemblata dal contesto in cui è calata, il quale ha la possibilità di decidere il comportamento senza dover conoscere i dati usati dagli algoritmi.

Permette inoltre di sostituire blocchi condizionali negli algoritmi spostando i casi in classi Strategy separate che ne contengono l'implementazione. Le suddette caratteristiche rendono il pattern un'alternativa al subclassing.

Il contesto deve tuttavia conoscere le classi che implementano gli algoritmi, così come le interfacce e i metodi, poiché è tramite questi che il contesto interagisce con gli algoritmi. Se le strategy devono offrire un'interfaccia comune, le comunicazioni con il contesto possono divenire eccessivamente complicate e alcuni dei dati passati dal contesto potrebbero restare inutilizzati. Alternativamente può essere la strategy a chiedere i dati al contesto, ma questo la lega all'interfaccia di quest'ultimo. In figura 4.1 è rappresentato schematicamente il pattern.

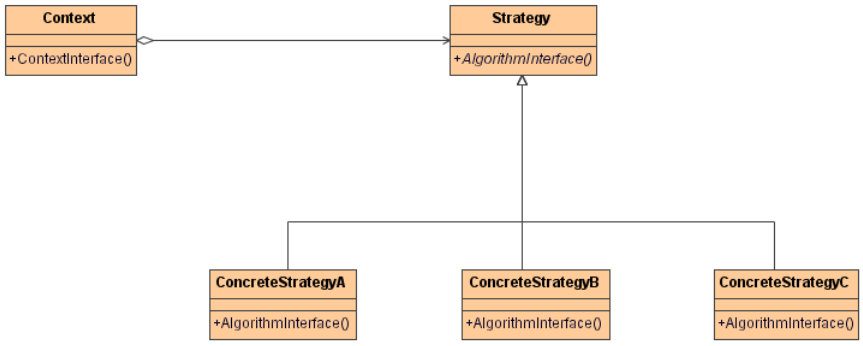


Figura 4.1: Il pattern strategy

Nella figura, Strategy definisce un'interfaccia comune a tutti gli algoritmi supportati. Context usa tale interfaccia per usufruire degli algoritmi definiti da ConcreteStrategy.

Esempio di applicazione

Come esempio citiamo il modo in cui è composta la classe che gestisce la visualizzazione. Come detto in precedenza, tale classe deve supportare diverse modalità di visualizzazione, variabili a seconda dei dati da visualizzare.

A questo scopo la classe viene modellata come composizione di classi Strategy, ognuna delle quali copre una parte della procedura di visualizzazione. Istanziando la classe si forniscono degli oggetti di tipo ConcreteStrategy, ognuno dei quali fornisce l'implementazione per la parte della procedura coperta dalla corrispondente Strategy. In questo modo è possibile limitare il subclassing alle ConcreteStrategy, che per il ruolo che ricoprono sono indipendenti l'una dall'altra.

Se si facesse ricorso all'ereditarietà anche per la classe di visualizzazione principale, andrebbe definita una classe base, ad esempio *View*. Questa andrebbe specializzata come:

- *ImageView* per la visualizzazione delle immagini;
- *3DView* per la visualizzazione delle rappresentazioni tridimensionali.

Supponiamo di voler visualizzare gli oggetti 3D con proiezione sia ortogonale che prospettica. L'abero delle classi verrebbe composto dalla classe base View, con due figli *ImageView* e *3DView*. Il codice per l'illuminazione sarebbe sicuramente replicato nelle foglie dell'albero, ottenute come risultato della combinazione delle funzionalità, a meno di usare ereditarietà multipla. Nella prima sezione del capitolo si è spiegato il motivo per cui si è preferito un approccio per composizione a uno basato su ereditarietà multipla. Qui ricordiamo brevemente solo che l'approccio scelto rende facile il riuso del codice, infatti il ricorso al pattern Strategy permette di definire la classe View come composizione di due oggetti, uno di tipo *Projection* e uno di tipo *GraphicsObject*, che forniscono l'interfaccia per accedere alle operazioni comuni. Derivando da Projection le classi

- *OrthoProjection*
- *PerspectiveProjection*

si forniscono le implementazioni specializzate per le proiezioni ortogonale e prospettica. Allo stesso modo si possono definire

- *ImageObject*
- *3DObject*

come estensioni di GraphicsObject. Con la gerarchia così definita è possibile fare qualsiasi combinazione senza replicazione di codice. Inoltre un'eventuale nuova classe che volesse far uso delle funzionalità di implementazione, potrebbe avvalersi delle classi già definite.

Quindi la definizione del comportamento di una classe di visualizzazione è ottenuta all'istanziazione dell'oggetto, ed è trasparente ai client che ne fanno uso. Avendo scelto il C++ come linguaggio per lo sviluppo, è stato possibile definire la classe base View come *template*, i cui parametri sono i tipi delle classi Strategy che la compongono. In questo modo è anche possibile specializzare il comportamento della classe a tempo di compilazione, mediante una tecnica nota come *template meta programming*.

Tornando al caso generale, va ricordato che la limitazione principale del pattern Strategy consiste nell'obbligo per le classi ConcreteStrategy di offrire la stessa interfaccia della superclasse.

4.2.2 Decorator

Il pattern Decorator permette di aggiungere responsabilità ad un oggetto, o di modificarne il comportamento dinamicamente senza far uso di subclassing. Dato un *component*, l'oggetto il cui comportamento deve essere modificato, si applica un decorator, la cui interfaccia deve essere conforme a quella dell'oggetto component. Tutte le richieste originariamente dirette al component sono gestite dal decorator, il quale può aggiungere del comportamento prima o dopo aver invocato le funzionalità del component. In questo modo il decorator risulta trasparente rispetto al client che usa il component e totalmente invisibile al component stesso.

Rispetto all'ereditarietà offre alcuni vantaggi:

- offre maggiore flessibilità, in particolare a run time, essendo possibile

cambiare dinamicamente il Decorator applicato ad un component e con esso il comportamento dell'oggetto;

- evita che nascano catene di ereditarietà troppo lunghe e complesse poiché sostituisce, almeno parzialmente, il meccanismo dell'ereditarietà nella specializzazione degli oggetti;
- permette di definire un component solo con i dati e le funzionalità minime, estendendolo con l'applicazione di oggetti Decorator;
- favorisce il riuso del codice, incapsulando funzionalità comuni a più oggetti nei Decorator.

In figura 4.2 è schematizzato un esempio di struttura per il pattern.

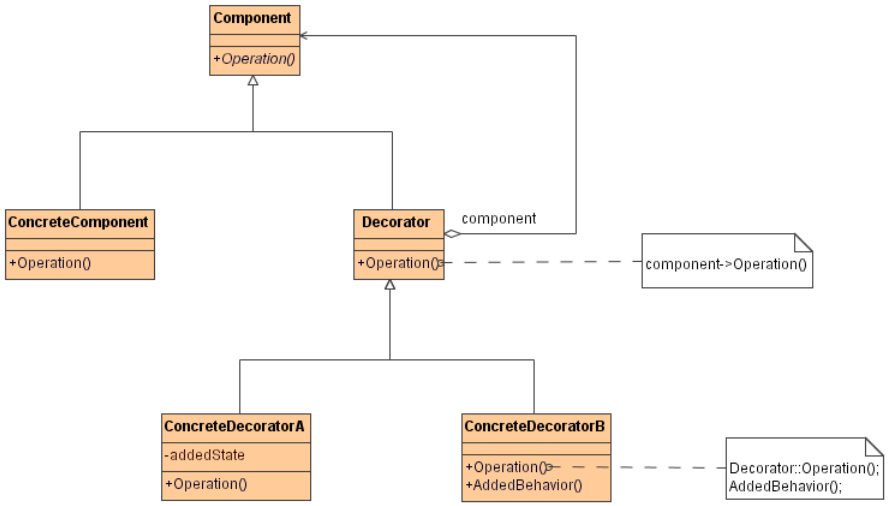


Figura 4.2: Il pattern decorator

Esempio di applicazione

Il pattern stato usato per rendere le procedure di disegno componibili. Per modificare il modo in cui un oggetto viene disegnato è sufficiente inserire decorator nella catena di visualizzazione, in modo che le operazioni di disegno attraversino prima il decorator e poi l'oggetto. Questo rende semplice

aggiungere o modificare particolari al disegno senza che sia necessario modificare l'oggetto da disegnare. Ad esempio l'applicazione dell'illuminazione agli oggetti 3D è effettuata con un oggetto Decorator. In qusto modo modo per cambiare il modello di illuminazione basta sostituire, anche dinamicamente, l'oggetto.

Decorator ha permesso inoltre di modellare il sistema di reazione agli input esterni in maniera simile a quanto fatto per la visualizzazione. Aggiungendo oggetti di tipo Decorator alla catena di reazione all'input si consente all'oggetto di intercettare l'input per eseguire delle operazioni. Se l'oggetto non intende reagire all'input propagherà semplicemente il messaggio lungo la catena. Ad esempio, la gestione del posizionamento dell'utente nel modello è modellata con Decorator, è così possibile attivarla e disattivarla in qualsiasi momento.

4.2.3 Command

Il pattern Command permette di incapsulare in una classe un comando insieme all'implementazione del comando stesso. In questo modo è possibile separare l'implementazione del comando da chi lo invoca, condividere lo stesso comando tra più unità invocanti e sostituire dinamicamente il comando. In figura 4.3 è rappresentato schematicamente il pattern.

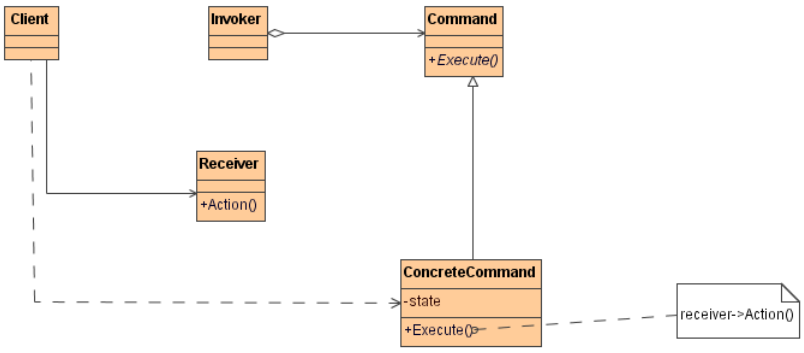


Figura 4.3: Il pattern Command

Un Command può avere un handle verso un receiver, al quale deve essere applicato il comando. Command offre solo l'interfaccia per la chiamata del

metodo *Execute*. Il client che crea il ConcreteCommand passa all'oggetto un riferimento al receiver creando il binding tra chiamata ed esecuzione.

I vantaggi più immediati che seguono dall'adozione di una classe Command sono:

- la possibilità di definire dei Command come macro per creare comandi come composizione di altri comandi;
- la facilità nell'aggiungere comandi estendendo la classe base;
- la possibilità di offrire a ogni comando un grado di intelligenza diverso. Diventa così possibile definire comandi che implementano una funzionalità in maniera del tutto autonoma, oppure comandi che eseguono solo il binding verso un metodo del receiver.

Esistono diversi accorgimenti che si possono usare per migliorare l'efficienza o la flessibilità di una classe Comand. Ad esempio se si usa il linguaggio C++ è possibile definire la classe command come *template*. Fornendo come parametro di template il tipo dell'oggetto reveiver si rende il comando applicabile a tutti gli oggetti che forniscono un metodo con una determinata segnatura. L'uso di *function object* in combinazione con i template può anche permettere di rendere la classe Command indipendente dal receiver e dal metodo da invocare. Il binding viene risolto all'istanziazione dell'oggetto Command.

Esempio di applicazione

Nella sezione 4.2.2 si è illustrato come l'interazione sia modificabile componendo una catena di reazione all'input con oggetti di tipo Decorator. Poiché la catena potrebbe contenere oggetti predisposti per reagire agli stessi eventi, è necessario fornire un modo per attivare e disattivare interattivamente tali oggetti. Questo è possibile utilizzando il pattern Command, assegnando a un oggetto di questo tipo un comando che permetta di attivare o disattivare degli oggetti Decorator.

Fornendo agli elementi più esterni dell'applicazione la lista dei Command abilitati, si espone la lista degli elementi di interazione attivabili. Questa

informazione si traduce in una lista di funzionalità da presentare all'utente dell'applicazione, il quale ha così la possibilità di attivare e disattivare gli elementi che compongono il modello di interazione.

Ad esempio l'attivazione o la disattivazione del gestore del posizionamento, è associata ad un Command. All'utente viene presentata la lista delle funzionalità associate ad una View ogni qualvolta preme il tasto destro del mouse nello spazio della View. Gli elementi della lista derivano dai command esposti dalla View. Se tra questi è presente il gestore del posizionamento, l'utente può attivarlo e passare alla modalità di interazione associata.

4.2.4 Facade

Il pattern Facade permette di fornire un'interfaccia semplice verso sottosistemi complessi. Le classi interne del sottosistema rimangono comunque accessibili, in modo da rendere possibile estendere o assemblare gli elementi del sottosistema in modo diverso da quello previsto dalla classe facade. Se un sistema è suddiviso in strati l'uso di classi facade permette di semplificare le comunicazioni tra gli strati.

In figura 4.4 è proposta un esempio di organizzazione che sfrutta una classe facade.

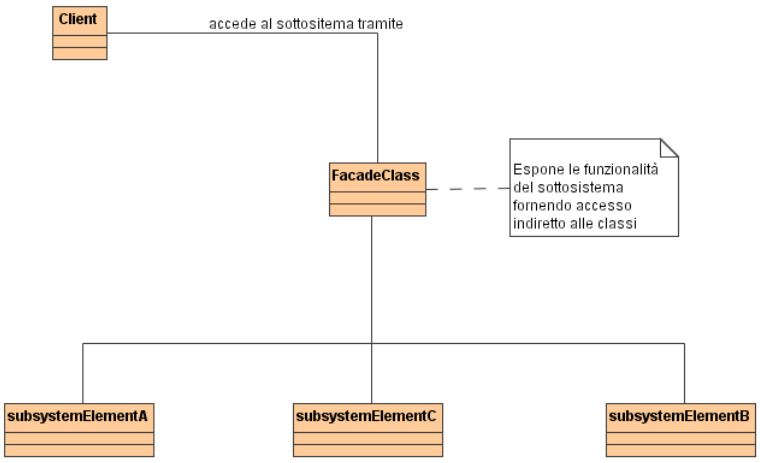


Figura 4.4: Il pattern Facade

L'uso di una classe di questo tipo permette di semplificare l'uso di un sottosistema e rende i client indipendenti dagli elementi di quest'ultimo. il fatto che i client diventino indipendenti dalle classi interne porta vantaggi anche dal punto di vista della programmazione. Permette infatti di ridurre i tempi di compilazione, eliminando la necessità di ricompilare un client quando variano solo le classi interne al sottosistema.

Permette ada esempio di richiedere la lettura di una lista di file, di organizzarne il contenuto e di esporne gli attributi. In questo modo è possibile richiedere la lettura di una determinata immagine, appartenente a uno studio e a una serie per precise, senza conoscere il gestore dello stream di input o il la classe che gestisce il dizionario dei degli attributi.

Esempio di applicazione

L'applicazione del pattern ha permesso di definire un'unica classe come punto di accesso semplificato al modulo di lettura dei file. Attraverso questa classe l'applicazione accede alle funzionalità del modulo. Più precisamente la classe Facade legge i file, ne estrae le informazioni e organizza i dati in base agli attributi prelevati. Per fare ciò si affida alle classi interne del modulo, ma poiché questo avviene in maniera del tutto trasparente, chi la usa non conosce i dettagli del funzionamento o le classi a cui si appoggia. Le classi interne restano ovviamente disponibili per chiunque volesse utilizzarle direttamente, senza far uso della classe ora descritta.

4.2.5 TemplateMethod

Definisce lo scheletro di un algoritmo in un'operazione di una classe, lasciando la definizione di alcuni passi dell'algoritmo alle sottoclassi. Permette alle sottoclassi di ridefinire alcuni passi di un algoritmo senza cambiarne la struttura. In figura 4.5 è mostrata schematicamente la struttura descritta.

Le operazioni che possono essere ridefinite vengono dette *operazioni primitive*. Possono essere virtuali pure o semplicemente virtuali. Nel caso di funzioni virtuali semplici queste forniscono un'implementazione standard, spesso vuota, che può essere ridefinita se necessario. Operazioni di questo tipo vengono dette *hook operations*.

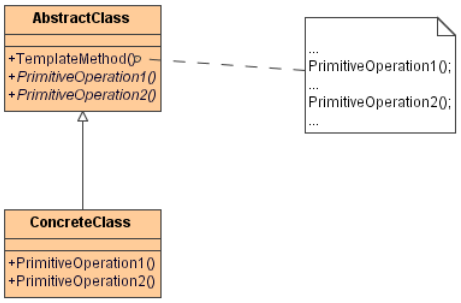


Figura 4.5: Il pattern Template Method

Esempio di applicazione

Nell'applicazione il pattern è stato usata nelle procedure di configurazione di oggetti complessi. Per tali oggetti è definita una classe base che fornisce l'interfaccia e un TemplateMethod per la costruzione dei componenti. Il TemplateMethod fissa dei passi per la configurazione, chiamando per ciascuno di questi una funzione virtuale. Le sottoclassi forniscono l'implementazione dei passi di configurazione.

4.2.6 Mediator

Definisce un oggetto che incapsula le comunicazioni tra un insieme di oggetti detti *colleague*. Permette di rendere gli oggetti indipendenti l'uno dall'altro eliminando la necessità che si riferiscano esplicitamente tra loro. Sarà infatti l'oggetto mediator a far transitare l'informazione da un colleague all'altro.

I principali vantaggi sono:

- limita il subclassing. Se cambia il modello di cooperazione basta specializzare il Mediator, i colleague possono essere usati senza alcuna modifica;

- semplifica la progettazione degli oggetti e ne favorisce il riuso poiché i colleague non sanno nulla l'uno dell'altro;

- rappresenta un'astrazione del modello di cooperazione, in fase di pro-

gettazione separa in modo chiaro il comportamento di un oggetto dalla sua cooperazione con gli altri.

Il mediator ha riferimenti verso i colleague, ognuno dei quali ha un riferimento verso il mediator. In questo modo un colleague può segnalare un evento, ad esempio l'avvenuta modifica di una parte dell'oggetto, per il quale il mediator ha una politica di gestione. Le politiche di gestione del mediator comprendono per esempio decidere quali sono i colleague da aggiornare e segnalare loro il cambiamento.

Esempio di applicazione

Mediator è stato applicato per semplificare al massimo le politiche di comunicazione tra gli elementi del sistema, e per rendere tali politiche facilmente modificabili. Ha permesso di incapsulare le comunicazioni degli oggetti che gestiscono la visualizzazione e l'interazione con l'oggetto che controlla gli accessi ai dati. Nessuno dei suddetti elementi conosce nulla degli altri; questo rende di fatto i componenti indipendenti l'uno dall'altro.

Più in generale le comunicazioni sono gestite nel modo seguente: quando un elemento del sistema vuole notificare agli altri un cambiamento di stato, fa una segnalazione all'oggetto Mediator, il quale deciderà, secondo le politiche da lui implementate, a quali elementi notificare il cambiamento. Inoltre potrà eseguire operazioni di trasformazione dell'informazione a seconda dell'interfaccia degli elementi che prendono parte alla comunicazione.

Quest'opera di centralizzazione delle comunicazioni nell'oggetto Mediator presenta sia vantaggi che svantaggi:

- rende facile tenere sotto controllo le comunicazioni;
- rende l'implementazione di un componente indipendente da quella di tutti gli altri;
- rende semplice la modifica delle politiche di comunicazione, essendo sufficiente estendere la classe Mediator usata;
- lega però il Mediator strettamente all'architettura del modulo nel quale è inserito, rendendone difficoltoso il riuso.

I vantaggi che Mediator comporta sono stati considerati sufficientemente importanti da poter tollerare l'unico svantaggio.

Un esempio d'uso concreto è costituito dalla procedura di caricamento di un dataset. Quando un elemento del sistema richiede al mediator il caricamento, questo inoltra la richiesta al gestore del dataset. Se il caricamento avviene correttamente, il mediator segnala l'evento agli elementi di visualizzazione che ritiene debbano essere aggiornati, fornendo i dati necessari per la visualizzazione.

Capitolo 5

Image Processing

In questo capitolo si accenneranno le caratteristiche delle immagini. Successivamente si introdurranno alcune tecniche di *Image Processing*, utili nella fase di preparazione alla ricostruzione tridimensionale.

5.1 Caratteristiche delle immagini

Osservando le immagini della colonscopia corrispondenti alle acquisizioni sul piano assiale, si nota come queste siano caratterizzate da tre principali strati, disposti in modo concentrico (vedi fig. 5.3). Lo strato più esterno risulta completamente nero e corrisponde ad assenza di segnale, ossia non deriva da misurazioni della macchina TAC. Più all'interno troviamo uno strato di colore più chiaro che corrisponde all'aria all'esterno dell'addome. Il terzo strato circoscrive l'addome e presenta valori di intensità molto più alti rispetto agli strati precedenti. L'addome costituisce l'area di interesse per l'elaborazione.

Il colon è una delle strutture all'interno dell'addome che durante l'esame vengono gonfiate con aria.

Poiché un'immagine rappresenta una sezione del volume essa sarà dotata di uno spessore. Un pixel rappresenta quindi il valore medio del segnale proveniente da un elemento del volume. Se nello spazio occupato da un voxel si trovano sia aria che tessuto, il valore calcolato sarà una media dei valori di intensità dell'aria e del tessuto. Questo comporta che in fase di elaborazione il pixel corrispondente possa non essere interpretato in modo

corretto. In particolare un pixel appartenente a un bordo potrebbe essere confuso con un pixel appartenente a una zona d'aria. Questo fenomeno è chiamato *effetto di volume parziale* o *partial volume effect* (vedi fig. 5.1).

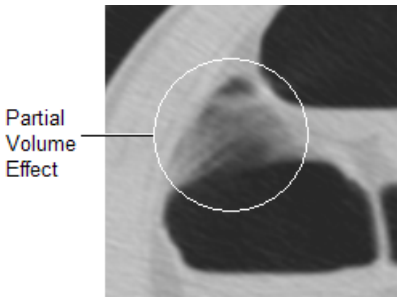


Figura 5.1: Esempio di partial volume effect

Per capire come questo fenomeno influisce sull'elaborazione si pensi a come varia l'intensità dei pixel se si attraversano una zona occupata esclusivamente da aria e successivamente una zona contenente tessuto. L'intensità passerà da valori prossimi allo zero a valori nettamente più alti. La rapidità di tale variazione si rivelerà caratteristica determinante per l'individuazione dei bordi. In caso di partial volume effect i valori variano in modo più lento e irregolare (vedi fig. 5.2).

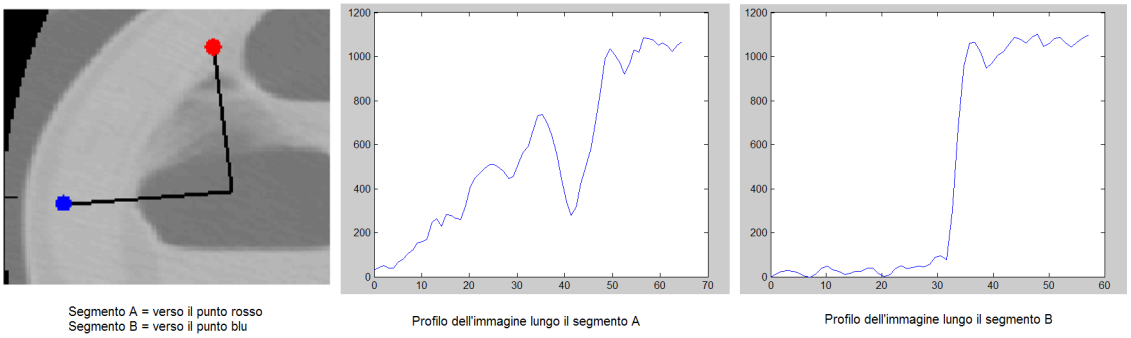


Figura 5.2: Valori dei pixel in presenza e in assenza di partial volume effect

L'aria è presente sia nella parte di volume esterna all'addome che all'interno del lume e di altre strutture cave (per esempio i polmoni). Questo

comporta che i voxel all'esterno dell'addome abbiano valori di intensità del tutto analoghi a quelli del colon. Pertanto al fine di semplificare l'individuazione, o segmentazione, del colon, sarà necessario rimuovere la zona d'aria all'esterno dell'addome, che nel seguito sarà indicata come *background*.

Alle immagini saranno inoltre applicati dei filtri per la rimozione del rumore.

Di seguito vengono descritte le tecniche e gli algoritmi usati per la rimozione del background e il filtraggio dell'immagine.

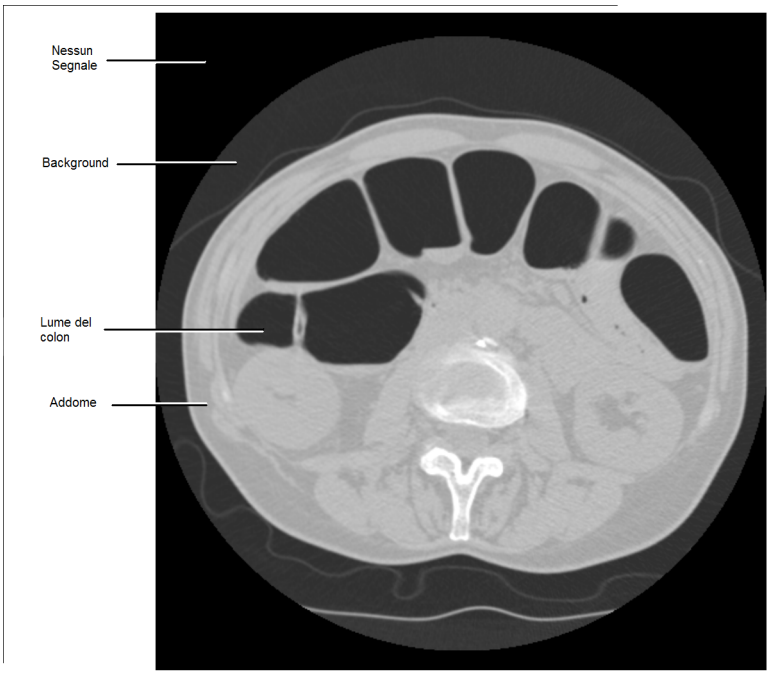


Figura 5.3: Background di un'immagine assiale

5.2 Rimozione del background

La rimozione del background è stata effettuata applicando alcuni operatori elementari della morfologia matematica.

La *morfologia matematica*, introdotta alla fine degli anni '60, è uno strumento molto potente per l'analisi non lineare delle immagini [Mat75, Ser82,

Ser87, Vin95, Roe96]. Metodi morfologici pongono l'accento sulla *forma* degli oggetti presenti nelle immagini, aspetto che li distingue nettamente dagli algoritmi classici, come i filtri descritti in questo capitolo, che sono più vicini al calcolo matematico puro.

Una descrizione esaustiva della morfologia matematica non rientra negli scopi di questa tesi, di conseguenza nei prossimi paragrafi verrano fornite solo le definizioni degli operatori morfologici usati negli algoritmi implementati.

Operatori morfologici binari

La morfologia matematica è stata inizialmente introdotta per operare su immagini binarie e solo successivamente i suoi operatori sono stati estesi per trattare livelli di grigio.

Un'immagine binaria può essere considerata un insieme di punti bidimensionali. Ad esempio l'immagine X in figura 5.4 coincide con l'insieme:

$$X = \left\{ (0,0)\,,(1,1)\,,(2,2)\,,(3,1) \right\}$$

Una *trasformazione morfologica* Ψ è definita come una relazione tra l'imma-

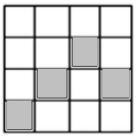


Figura 5.4: Esempio di immagine binaria ≡ insieme di punti

gine (intesa come insieme di punti) ed un piccolo insieme di punti B chiamato *elemento strutturale*, espresso in funzione di una origine locale $\mathcal{O}$. L'applicazione della trasformazione morfologia all'immagine $\Psi(X)$ consiste nel muovere l'elemento strutturale sull'intera immagine X. L'origine $\mathcal{O}$ di B viene posta su ogni punto di X ed il risultato della relazione viene salvato in una nuova immagine X'. La figura 5.5 mostra forme tipiche degli elementi strutturali.

Sia $E = \mathbb{Z}^2$, X l'immagine binaria e B l'elemento strutturale. La *dilatazione* $\oplus$ combina i due insiemi X e B tramite una semplice addizione tra

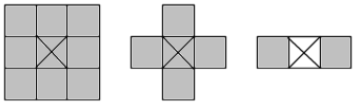


Figura 5.5: Possibili elementi strutturali

vettori:

$$\Psi_\delta(X) = X \oplus B = \left\{ p \in E |\, p = x + b, \quad x \in X,\, b \in B \right\}$$

La dilatazione tramite elementi strutturali che contengono l'origine $\mathcal{O}$ è una trasformazione isotropica, in quanto ha lo stesso effetto in tutte le direzioni.

La dilatazione può essere descritta come una trasformazione che:

- riempie piccoli "buchi" all'interno di X;
- trasforma tutti i pixel dello sfondo adiacenti al bordo di un oggetto in pixel appartenenti all'oggetto;

Se l'origine $\mathcal{O}$ non è inclusa in B, il risultato della dilatazione può cambiare radicalmente l'immagine: tutti gli elementi strutturali usati nella tesi contengono l'origine locale.

L'*erosione* $\ominus$ è definita come:

$$\Psi_\epsilon(X) = X \ominus B = \left\{ p \in E |\, \forall b \in B.\, p + b \in X \right\}$$

L'erosione può essere descritta come una trasformazione che:

- cancella tutti gli oggetti in X più piccoli dell'elemento strutturale;
- trasforma tutti i pixel del contorno degli oggetti maggiori di B in pixel dello sfondo;

L'erosione e la dilatazione non sono, in generale, operazioni inverse.

La combinazione delle due trasformazioni crea due importanti trasformazioni morfologiche: l'apertura (opening) e la chiusura (closing).

L'*apertura* $\circ$ viene definita come una erosione seguita da una dilatazione con lo stesso elemento strutturale:

$$\Psi_o(X) = X \circ B = (X \ominus B) \oplus B$$

L'apertura cancella gli oggetti con un'area inferiore a quella di B, conservando tutti gli elementi con un'area maggiore;

La *chiusura* $\bullet$ è definita come una dilatazione seguita da una erosione con lo stesso elemento strutturale:

$$\Psi_c(\mathsf{X}) = \mathsf{X} \bullet \mathsf{B} = (\mathsf{X} \oplus \mathsf{B}) \ominus \mathsf{B}$$

La chiusura riempie i buchi (holes) di X con un'area inferiore a quella dell'elemento strutturale.

I bordi esterni di un oggetto più grande di B possono essere calcolati nel seguente modo:

$$\text{Bordi}(\mathsf{X}) = (\mathsf{X} \oplus \mathsf{B}) - \mathsf{X}$$

oppure, alternativamente:

$$\text{Bordi}(\mathsf{X}) = \mathsf{X} - (\mathsf{X} \ominus \mathsf{B})$$

Nel primo caso il bordo è formato da pixel esterni all'oggetto, nel secondo da pixel appartenenti all'oggetto.images/segm/ridlerThresh1

In alcuni casi è utile restringere ad una sottoregione dell'immagine l'effetto di trasformazioni morfologiche. Per restringere gli effetti di un'erosione alla zona M dell'immagine X si usa l'*erosione condizionale*, così definita:

$$\Psi_\epsilon(\mathsf{X}|\mathsf{M}) = (\mathsf{X} \ominus \mathsf{B}) \cap \mathsf{M}$$

Analogamente viene definita la *dilatazione condizionale*:

$$\Psi_\delta(\mathsf{X}|\mathsf{M}) = (\mathsf{X} \oplus \mathsf{B}) \cap \mathsf{M}$$

Algoritmo implementato

L'algoritmo implementato, descritto nel prossimo listato, rimuove il background dalle immagini. Usa l'operatore morfologico erosione.

Sia $\mathcal{I}$ l'immagine a cui va rimosso il background, l'algoritmo implementato è il seguente:

Algoritmo 5.1: Segmentazione addome/sfondo

Input: Immagine di input $\mathcal{I}$;

Output: Immagine di ouput $\mathcal{I}_o$ ottenuta da $\mathcal{I}$ dando a tutti i voxel dello sfondo un valore pari al Padding Value specificato del file DICOM di origine;
immagine binaria $\mathcal{I}_b$, i cui voxel posti a 1 corrispondono all'addome, quelli posti a 0 allo sfondo;

1: $\mathcal{H} \leftarrow$ computeHistogram($\mathcal{I}$);
2: $\theta \leftarrow$ findThreshold($\mathcal{H}, \lambda$); // Calcola un valore di soglia
3: $\mathcal{I}_\theta \leftarrow$ applyThreshold($\mathcal{I}$, θ); // Produce una maschera binaria
4: $\mathcal{I}_E \leftarrow (\mathcal{I}_\theta \ominus B)$; // Erosione della maschera binaria
5: $\mathcal{I}_{edges} \leftarrow \mathcal{I}_\theta - \mathcal{I}_E$;
6: $\mathcal{R} \leftarrow$ extractRegions($\mathcal{I}_{edges}$);
7: **for all** $R \in \mathcal{R}$ **do**
8: $\quad R \leftarrow$fillHoles(R);
9: **end for**
10: Inizializza $\mathcal{I}_b$ ponendo a 0 tutti i voxel;
11: **for all** $R \in \mathcal{R}$ **do**
12: $\quad$ **for all** $p \in R$ **do**
13: $\quad\quad \mathcal{I}_b(p) = 1$;
14: $\quad$ **end for**
15: **end for**
16: $\mathcal{I}_o \leftarrow$ applyMask($\mathcal{I}_b$);

Il comando 1 calcola l'istogramma dell'immagine. Con il comando 2 viene calcolata a partire dall'istogramma una soglia che separi chiaramente i voxel del background da quelli dell'addome. Il calcolo della soglia è molto semplice ma dà buoni risultati. Se $\mathcal{P}_1$ è il primo picco dell'istogramma e $\mathcal{P}_2$ l'ultimo, la soglia θ viene calcolata come $\theta = \lambda * (\mathcal{P}_1 + \mathcal{P}_2)$. Ponendo $\lambda = 0.55$ si ottiene una soglia che costituisce sicuramente un limite superiore per i valori dei voxel del background. In figura 5.6 è rappresentato un esempio di istogramma. Sono evidenziati il picco $\mathcal{P}_1$, il picco $\mathcal{P}_2$ e la soglia θ.

Il comando 3 consiste nell'applicare la soglia appena calcolata all'immagine originale. Come risultato si ottiene una maschera binaria $\mathcal{I}_\theta$ dove un pixel ha valore 1 se il pixel corrispondente dell'immagine originale ha valore

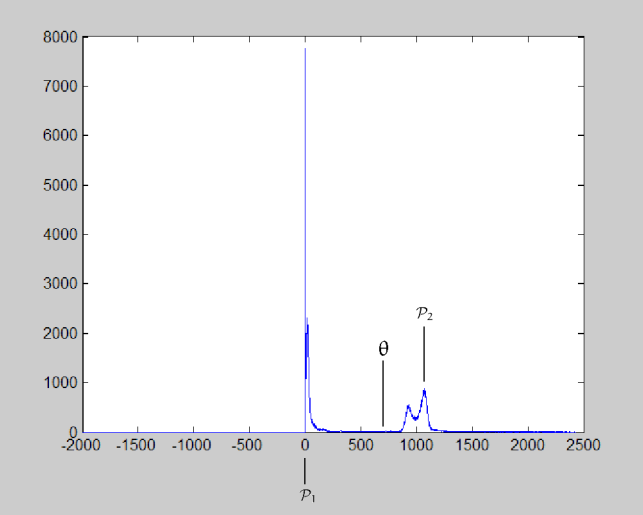


Figura 5.6: Misurazione della soglia nell'istogramma

maggiore della soglia, 0 altrimenti.

Per ottenere il bordo dell'addome si effettua l'erosione, nel comando 4, della maschera $\mathcal{I}_\theta$ producendo una maschera $\mathcal{I}_\mathsf{E}$. Successivamente si crea una nuova maschera $\mathcal{I}_{\text{edges}}$ sottraendo $\mathcal{I}_\mathsf{E}$ da $\mathcal{I}_\theta$. Con la procedura *extractRegions* si verifica che i bordi ottenuti non contengano interruzioni.

Vengono quindi riempite le regioni con la procedura *fillHoles* e viene prodotta una maschera dove i pixel con valore 1 rappresentano pixel dell'addome. Infatti qualsiasi altra regione possa essere stata generata è comunque contenuta in quella che descrive l'addome. L'operazione di Hole Filling è implementata con operatori morfologici nel modo seguente. Data l'immagine contenente il contorno di una regione $\mathcal{I}_{\text{edges}}$, i buchi presenti possono essere riempiti con:

$$\text{FILL}(\mathcal{I}_{\text{edges}}) = \Psi_\delta(\mathsf{P}|\mathcal{I}^\mathsf{c}_{\text{edges}})$$

dove P è un'immagine binaria contenente un solo pixel all'interno del contorno in $\mathcal{I}_{\text{edges}}$. L'elemento strutturale usato è di tipo "a croce" (quello al centro della figura 5.5).

La maschera viene infine applicata all'immagine in modo da lasciare inalterati i pixel dell'addome e dando ai pixel del background un valore corrispondente al padding value specificato nel file DICOM di origine.

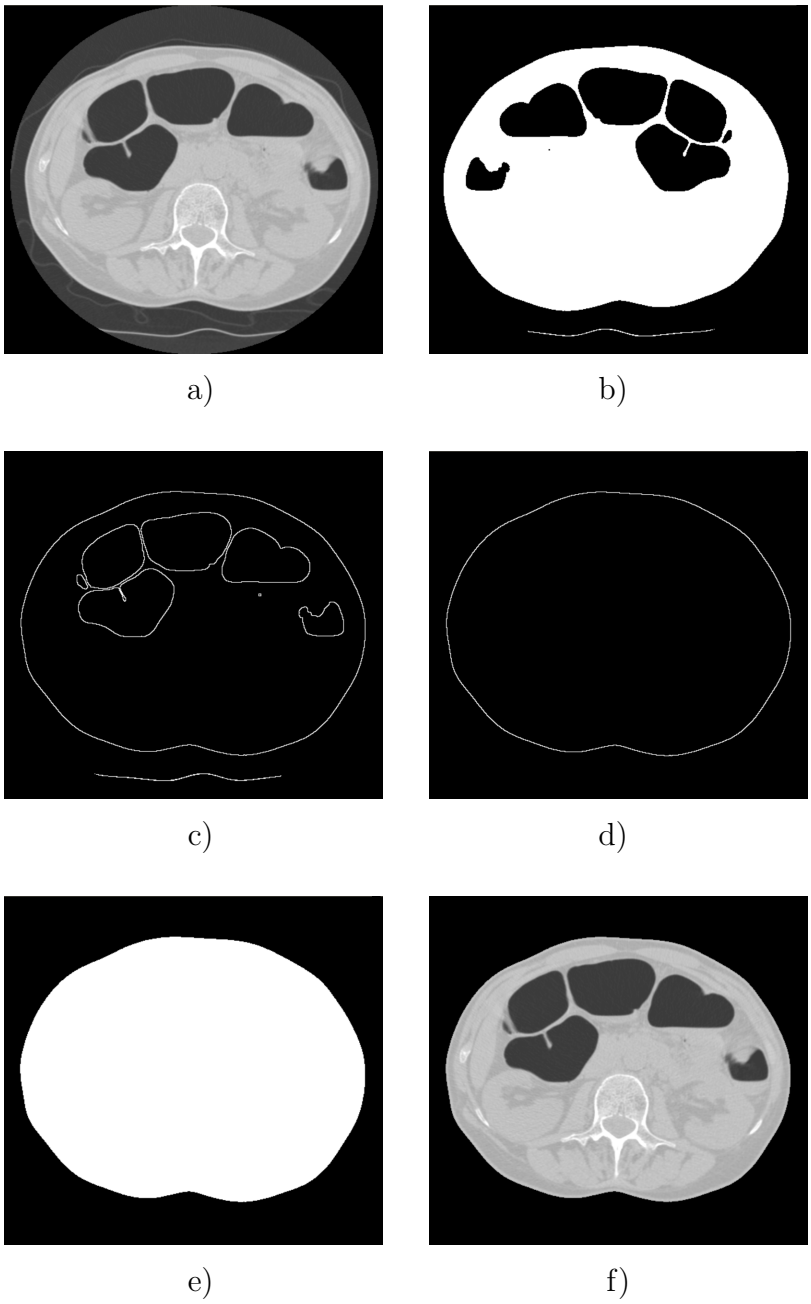


Figura 5.7: a) immagine originale; b) dopo l'applicazione della soglia; c) dopo la sottrazione; d) dopo aver individuato la regione principale; e) dopo holeFilling; f) background rimosso

5.3 Riduzione del rumore

In questa sezione saranno discussi i problemi legati alla presenza di rumore nelle immagini. Successivamente si presenteranno degli algoritmi di filtraggio per la rimozione del rumore; in particolare si introdurranno un filtro lineare e un filtro anisotropo.

5.3.1 Influenza del rumore sull'elaborazione

Il rumore presente nelle immagini può influenzare la costruzione del modello tridimensionale delle strutture.

Per quanto riguarda il primo aspetto si pensi alla corrispondenza tra il bordo di una struttura e i pixel che lo rappresentano nell'immagine. Se lungo tale bordo si trovano pixel riconducibili a rumore ma con intensità simile a quella dei pixel realmente appartenenti al bordo, questi verranno probabilmente considerati come facenti parte del bordo. Quest'ultimo risulterà quindi frastagliato, caratteristica che si ripercuoterà negativamente sulla qualità della rappresentazione tridimensionale (si veda 6.1).

In figura 5.8 è rappresentata un'immagine in cui le aree più scure corrispondono a zone dell'addome piene d'aria. L'immagine è stata prodotta selezionando dei valori di intensità per i pixel tali da mettere in risalto la presenza di rumore nelle zone d'aria. Si può notare come tali zone siano caratterizzate da valori non omogenei. Il rumore è riconoscibile come striature che occupano parti di volume che, in assenza di rumore, sarebbero visualizzate con livello di grigio pari a zero. Tale infatti è il fattore di attenuazione che caratterizza l'aria.

Per eliminare questi ed altri fattori di disturbo, le immagini subiscono alcune fasi di elaborazione prima che algoritmi di analisi o di ricostruzione tridimensionale vengano ad esse applicati.

5.3.2 Filtraggio dell'immagine

La necessità di eliminare o meglio di attenuare tale rumore ha portato a considerare e quindi implementare degli algoritmi di filtraggio, o *smoothing*, che costituissero una fase di pre-processing alla ricostruzione 3D.

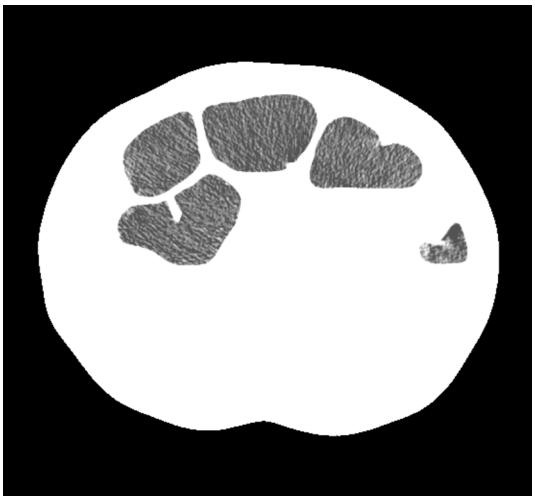


Figura 5.8: Esempio di rumore in un'immagine

Gli obiettivi della fase di smoothing, o, meglio, le caratteristiche dello smoothing ideale, sono [GKKJ92]:

1. minimizzare la perdita di informazione che può essere estratta da contorni netti e strutture molto dettagliate;

2. rimuovere efficientemente il rumore che corrompe regioni con proprietà fisiche omogenee;

3. evidenziare la definizione morfologica delle strutture;

In altre parole un filtro dovrebbe produrre una immagine in cui le regioni siano caratterizzate da una maggiore omogeneità, senza che questo comporti la perdita dell'informazione legata a bordi netti. Con questo scopo sono stati sperimentati due diversi filtri: ul filtro "media" e uno basato sulla diffusione anisotropa.

Filtro lineare

Il primo metodo sperimentato, per la sua semplicità, è un filtro lineare, detto *media*. Il filtro sostituisce ad ogni punto il valore medio dei pixel in un suo intorno prefissato (Figura 5.9).

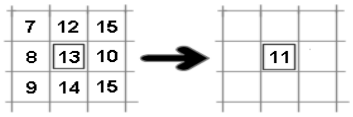


Figura 5.9: Average Filter: Media in un intorno 3×3

Il valore medio viene calcolato attraverso un prodotto di convoluzione tra la maschera di convoluzione (nel caso 3×3)

$$h = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

e l'immagine originale f, ripetendo la convoluzione per ogni pixel. L'immagine filtrata g, della stessa dimensione di f sarà:

$$g(i,j) = \sum_{m,n} \sum_{\in \mathcal{N}(i,j)} h(i-m, j-n) f(m,n) \tag{5.1}$$

dove con $\mathcal{N}(i,j)$ si intende l'insieme dei punti nell'8-vicinato del pixel $p = (i,j)$.

Le immagini prodotte appaiono "sfocate" in quanto il filtro, non essendo in grado di modificare la sua risposta in base al contesto spaziale della regione elaborata, annulla bordi leggeri e smussa eccessivamente bordi molto marcati che potrebbero rappresentare il confine tra regioni anatomiche differenti. In figura 5.10 è mostrato il risultato dell'applicazione del filtro sull'immagine in figura 5.8 con uno e poi tre passate e l'effetto di sfocamento provocato.

Filtro anisotropo

L'algoritmo di filtraggio anisotropo, introdotto in [PM87, PM90], può essere descritto come un processo di diffusione dei valori di grigio che effettua uno smoothing all'interno di una regione omogenea, conservando, allo stesso tempo, i bordi tra regioni vicine. L'immagine viene vista come un mezzo attraverso il quale un fluido si diffonde in base alle caratteristiche locali.

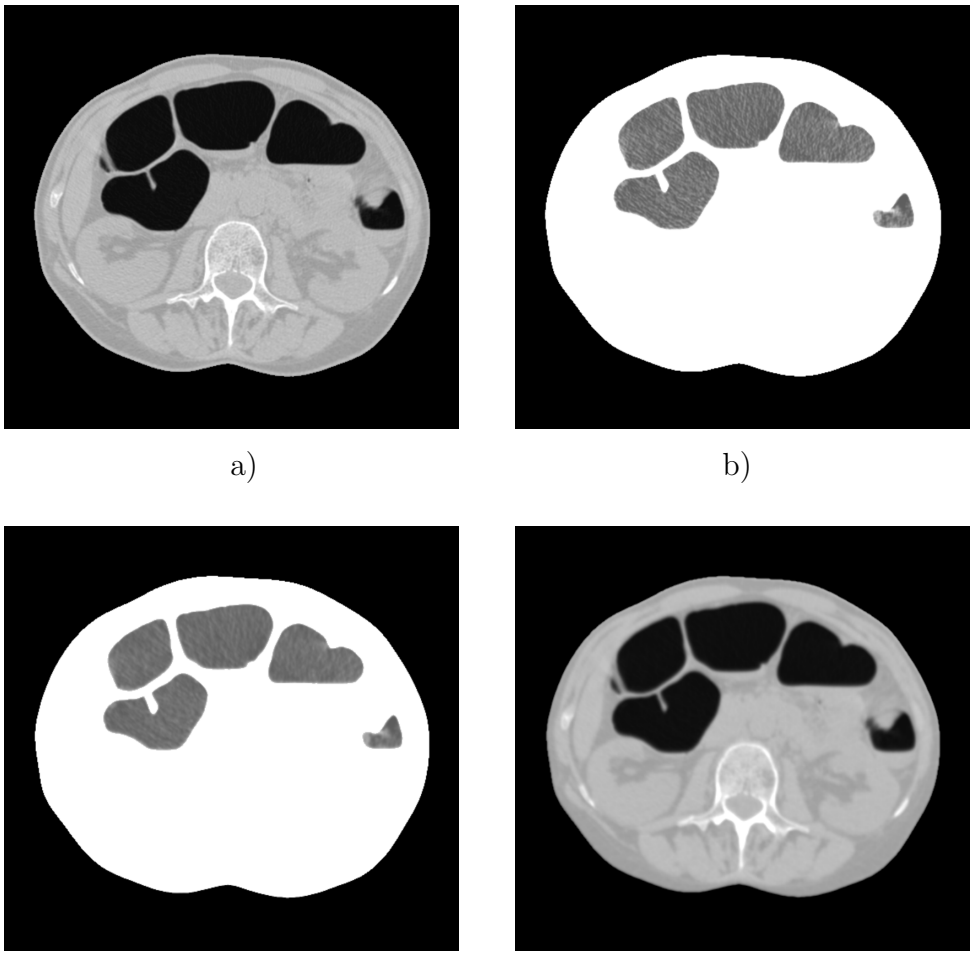


Figura 5.10: Esempi di applicazione dell'Average Filter. a) immagine originale; b) effetto di un passo di smoothing; c) effetto di tre passi di smoothing; d) effetto di sfocamento sull'immagine originale dopo tre passi di smoothing

L'idea alla base del metodo è quella di formare zone omogenee all'interno della stessa regione permettendo al processo di diffusione di superare bordi con un basso modulo. La diffusione del mezzo viene invece impedita in caso di bordi molto intensi, che , in genere, separarano regioni anatomiche differenti. Le pareti del colon sono caratterizzate da valori di intensità tale da permettere l'applicazione di un filtro anisotropo in maniera abbastanza aggressiva, senza che i bordi vengano modificati significativamente. Ciò con-

sente di ottenere una buona attenuazione del rumore all'interno del lume mantenendo intatte le caratteristiche delle pareti.

Il nome del metodo deriva dalla sua caratteristica di assegnare un peso diverso alle componenti del flusso provenienti da direzioni differenti.

Il processo di diffusione anisotropa è formulato come:

$$\frac{\partial}{\partial t}\mathcal{I}(p,t) = \mathrm{div}(c(p,t)\nabla\mathcal{I}(p,t)) \tag{5.2}$$

dove t rappresenta il parametro di ordine del processo oppure l'iterazione nel caso discreto, $p = (x, y)$ è un pixel/voxel dell'immagine e div è l'operatore *divergenza*. L'operatore divergenza è definito come:

$$div(A) = [\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}] \cdot A$$

dove A rappresenta un vettore di dimensione n.

La forza della diffusione è controllata dalla funzione $c(p,t)$, che impedisce la diffusione oltre bordi molto marcati. $c(p)$ deve essere una funzione monotona decrescente del gradiente dell'immagine, cioè:

$$c(p,t) = f(|\nabla\mathcal{I}(p,t)|), \qquad \text{monotona decrescente}$$

Le due forme più comuni che assume sono le seguenti:

$$c_1(p,t) = \exp\left(-\left(\frac{|\nabla\mathcal{I}(p,t)|}{K}\right)^2\right) \tag{5.3}$$

$$c_2(p,t) = \frac{1}{1 + \left(\frac{|\nabla\mathcal{I}(p,t)|}{K}\right)^{1+\alpha}} \qquad \alpha > 0 \tag{5.4}$$

Durante la sperimentazione sono state sperimentate entrambe, senza notare significative differenze. Nel resto del paragrafo si pone $c(p,t) = c_1(p,t)$, il grafico della funzione è tracciato in figura 5.11, il grafico del flusso in figura 5.12.

La regola per l'aggiornamento dei pixel può essere ricavata espandendo la formula (5.2) nello spazio discreto dell'immagine. Nella derivazione è preferibile semplificare la notazione ponendo $c(p,t) = c_t(p)$ e $\mathcal{I}(p,t) = \mathcal{I}_t(p)$.

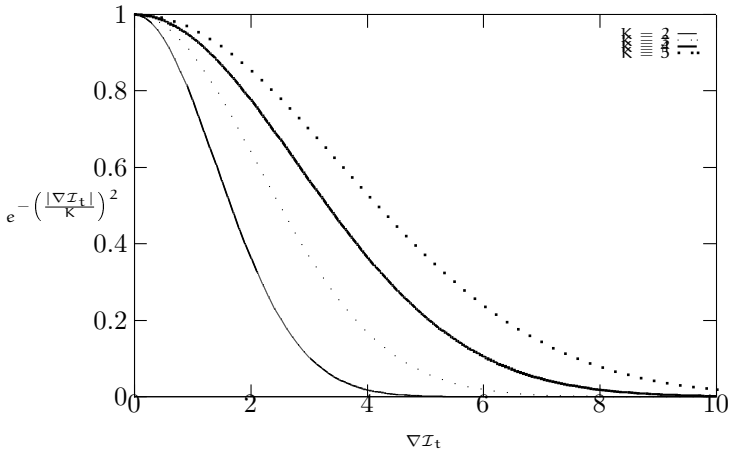


Figura 5.11: Funzione di diffusione $c_1(p,t)$

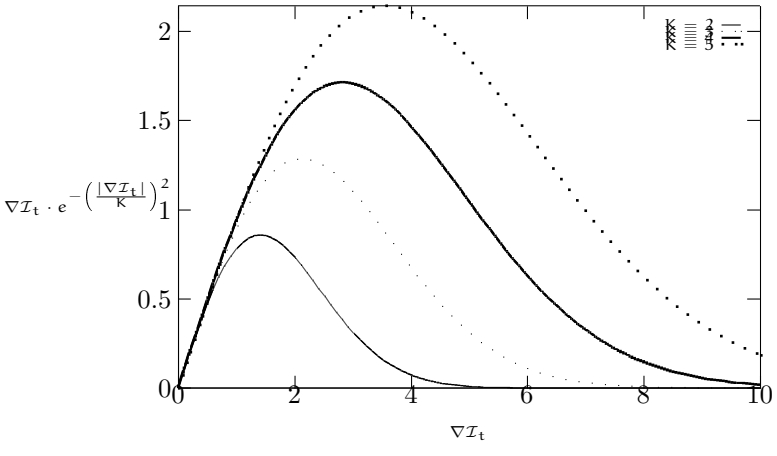


Figura 5.12: Grafico della diffusione in funzione di K

Sia $p = (x, y)$, allora:

$$\begin{aligned}
\frac{\partial}{\partial t}\mathcal{I}_t(p) =& \operatorname{div}(c_t(p)\nabla\mathcal{I}_t(p)) = \nabla^T(c_t(p)\nabla\mathcal{I}_t(p)) \\
=& \frac{\partial}{\partial x}\Big(c_t(p)\frac{\partial}{\partial x}\mathcal{I}_t(p)\Big) + \frac{\partial}{\partial y}\Big(c_t(p)\frac{\partial}{\partial y}\mathcal{I}_t(p)\Big)
\end{aligned}$$

$$\Big\{\text{approssimando le derivate con differenze su voxel distanti } \Delta x,\, \Delta y\Big\}$$

$$\begin{aligned}
\approx& \frac{1}{\Delta x}\left[c_t(x+\frac{\Delta x}{2}, y)\cdot\frac{\mathcal{I}_t(x+\Delta x, y)-\mathcal{I}_t(x,y)}{\Delta x} - c_t(x-\frac{\Delta x}{2}, y)\cdot\frac{\mathcal{I}_t(x,y)-\mathcal{I}_t(x-\Delta x, y)}{\Delta x}\right] + \\
& \frac{1}{\Delta y}\left[c_t(x, y+\frac{\Delta y}{2})\cdot\frac{\mathcal{I}_t(x, y+\Delta y)-\mathcal{I}_t(x,y)}{\Delta y} - c_t(x, y-\frac{\Delta y}{2})\cdot\frac{\mathcal{I}_t(x,y)-\mathcal{I}_t(x, y-\Delta y)}{\Delta y}\right] \\
=& \frac{1}{\Delta x^2}\left[\overbrace{c_t(x+\frac{\Delta x}{2}, y)\Big(\mathcal{I}_t(x+\Delta x, y)-\mathcal{I}_t(x,y)\Big)}^{\Phi_{est}} - \overbrace{c_t(x-\frac{\Delta x}{2}, y)\Big(\mathcal{I}_t(x,y)-\mathcal{I}_t(x-\Delta x, y)\Big)}^{\Phi_{ovest}}\right] + \\
& \frac{1}{\Delta y^2}\left[\overbrace{c_t(x, y+\frac{\Delta y}{2})\Big(\mathcal{I}_t(x, y+\Delta y)-\mathcal{I}_t(x,y)\Big)}^{\Phi_{nord}} - \overbrace{c_t(x, y+\frac{\Delta y}{2})\Big(\mathcal{I}_t(x,y)-\mathcal{I}_t(x, y-\Delta y)\Big)}^{\Phi_{sud}}\right]
\end{aligned}$$

Approssimando il gradiente con $\Delta x = \Delta y = 1$:

$$\frac{\partial}{\partial t}\mathcal{I}_t(p) = \Phi_{est} - \Phi_{ovest} + \Phi_{nord} - \Phi_{sud} \tag{5.5}$$

Una volta calcolato il flusso proveniente dalle quattro direzioni, l'immagine viene modificata secondo la seguente formula:

$$\begin{aligned}
\mathcal{I}_{t+\Delta t} =& \mathcal{I}_t + \Delta t\cdot\frac{\partial\mathcal{I}_t}{\partial t} \\
=& \mathcal{I}_t + \Delta t\cdot\big(\Phi_{est} - \Phi_{ovest} + \Phi_{nord} - \Phi_{sud}\big)
\end{aligned} \tag{5.6}$$

Il metodo descritto usa per il calcolo del flusso una relazione di 4-vicinato. Il calcolo può essere comunque esteso ad una relazione di 8-vicinato, dove viene calcolato anche il flusso proveniente dai vicini in diagonale: è sufficiente scalare le nuove componenti per la distanza dal pixel centrale, che non è più unitaria ma $\sqrt{2}$. Nell'algoritmo implementato, permette di calcolare il flusso secondo il 4-vicinato.

Il problema della convergenza del metodo è stato studiato successivamente in [Nor90]. La convergenza non è un problema in questa tesi in quanto

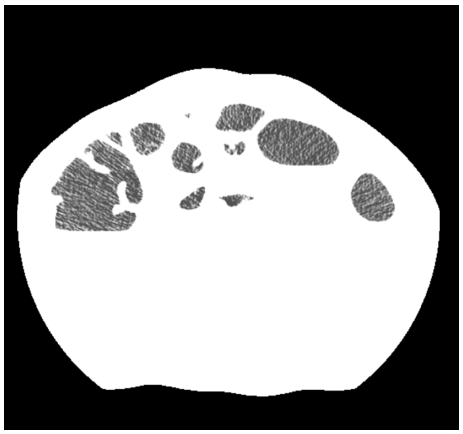
l'algoritmo viene interrotto dopo un numero finito di iterazioni stabilito, per tutte le immagini, a priori. L'unico parametro rimasto libero è la costante di aggiornamento $\Delta t > 0$. In [GKKJ92] il parametro viene calcolato in modo che garantisca la convergenza del metodo; l'analisi stabilisce che il valore massimo che esso può assumere è:

$$\Delta t_{max} = \frac{1}{1 + \sum_{q \in \mathcal{N}(p)} \frac{1}{\|q-p\|_2}}$$

quindi $\Delta_t \in [0, \frac{1}{5}]$ in caso di 4 vicinato oppure $\Delta_t \in [0, \frac{1}{7}]$ in caso di 8 vicinato.

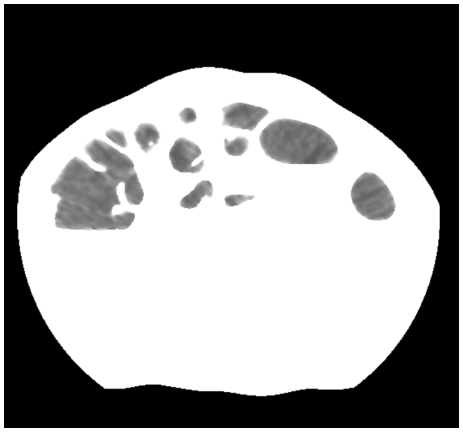
Come si nota dall'immagine 5.13 , il filtro anisotropo riesce a rendere le regioni molto omogenee preservando i bordi esistenti.

a)



b)

c)



d)

e)

Figura 5.13: Esempio di applicazione del filtro anisotropo. a) immagine originale; b) immagine che mette in evidenza il rumore; c) rumore dopo 5 iterazioni di filtro anisotropo; d) rumore dopo 10 iterazioni di filtro anisotropo; e) immagine originale filtrata con 10 iterazioni di filtro anisotropo

Capitolo 6

Ricostruzione 3D

Questo capitolo illustra come si è costruita una rappresentazione tridimensionale del colon a partire dai dataset volumetrici. Verranno brevemente introdotti i concetti fondamentali della visualizzazione 3D interattiva. Successivamente si decriverà la tecnica di ricostruzione adottata accennando alle alternative che si è deciso di scartare. Infine verranno illustrate le fasi di elaborazione applicate per migliorare la rappresentazione 3D.

6.1 Costruzione di un colon virtuale

L'esigenza di proporre una rappresentazione tridimensionale del colon nasce dalla difficoltà di consultazione dei dataset biomedici nel modo tradizionale. Il modo classico per ispezionare un volume di questo tipo consiste nell'osservare una per una le immagini che rappresentano le sezioni del volume. Questo tipo di interazione con il volume rende difficile "orientarsi" all'interno del colon, a causa delle caratteristiche anatomiche (vedi sez. 2.1.1).

Affiancare la visualizzazione 3D di un modello tridimensionale del colon, alla classica visualizzazione delle immagini, comporta i seguenti vantaggi:

- permette di avere una visione d'insieme della struttura complessiva del colon;
- permette di porre il punto di vista dell'osservatore all'interno della struttura, in modo da osservarne i particolari da diverse angolazioni;

- permette un'interazione più complessa e sofisticata con i dati, che deriva dal combinare l'interazione con le immagini e l'interazione con il modello 3D.

I primi due aspetti possono modificare radicalmente le modalità di ispezione del dataset, semplificandola e rendendola più rapida.

Attualmente le principali tecniche usate nella visualizzazione 3D sono due: *Surface Rendering* e *Volume Rendering*. Per questa tesi si è scelto di affidarsi a tecniche del primo tipo. Verranno ora descritte brevemente le due tecniche con i loro vantaggi e svantaggi, alla luce delle considerazioni fatte si motiverà la scelta della tecnica surface rendering.

Surface Rendering

Si basa sulla modellazione degli oggetti da visualizzare come maglie tridimensionali di poligoni, dette *mesh*. La rappresentazione classica è basata su mesh di triangoli. La visualizzazione degli oggetti modellati avviene calcolando il valore di ciascun pixel dello schermo come proiezione dei punti dei triangoli che compongono la mesh. Per rendere più realistica la visualizzazione possono essere applicate fonti di illuminazione, che permettono di attribuire un colore ad ogni vertice del modello, e di conseguenza ad ogni pixel calcolato, in base alla posizione e all'orientazione del modello rispetto alle fonti di luce. I modelli possono essere inoltre resi più realistici mediante l'uso di *texture*. Le texture sono immagini che vengono applicate ai poligoni per caratterizzarne la superficie.

Il rendering di mesh è la tecnica più diffusa per la visualizzazione 3D interattiva. I punti di forza del possono essere così riassunti:

- offre un'astrazione della geometria dell'oggetto, che può essere utile per elaborazioni sulla morfologi dell'oggetto stesso;
- è una tecnica che negli anni si è dimostata estremamente versatile;
- da la possiblità di sfruttare accelerazione hardware molto avanzata;
- la letteratura copre moltissimi aspetti e campi di utilizzo;

- esistono numerose librerie software dedicate.

I principali svantaggi sono:

- necessità di calcolare la rappresentazione poligonale degli oggetti da visualizzare;
- sensibile alla complessità della scena, intesa come numero di oggetti;
- sensibile alla complessità degli oggetti, intesa come numero di poligoni;
- non fornisce informazioni sullo spessore dell'oggetto.

Uno degli strumenti più importanti per lo sviluppo di applicazioni che fanno uso di questa tecnica è la libreria OpenGL. Un breve introduzione alla libreria è data in appendice B.

Volume Rendering

Si basa sulla visualizzazione diretta dei dati volumetrici che costituiscono un oggetto. Ogni pixel dello schermo è illuminato con il valore calcolato dalla proiezione dei voxel del volume. È una tecnica meno diffusa rispetto al rendering di superfici. I motivi della scarsa diffusione risiedono principalmente nei requisiti hardware. Tuttavia l'incremento di prestazioni che i dispositivi di visualizzazione hanno conosciuto negli ultimi anni ne sta agevolando la diffusione. Di seguito sono elencati i principali vantaggi:

- non è necessario calcolare una rappresentazione dei dati volumetrici;
- permette di visualizzare in modo naturale le caratteristiche di spessore degli oggetti;
- non risente della complessità degli oggetti e della scena.

Gli svantaggi sotto elencati costituiscono il principale freno alla diffusione della tecnica:

- è caratterizzata da ingente occupazione di memoria;
- fornisce un limitato sfruttamento dell'accelerazione hardware;

- il calcolo dell'illuminazione va programmato via software;
- comporta trasformazioni geometriche molto costose dal punto di vista del calcolo.
- ha un campo di utilizzo ristretto rispetto al rendering di superifici;
- scarse risorse in letteratura.

Per questa tesi si è scelto di affidarsi al rendering di superfici. Il motivo principale risiede nei requisiti hardware inferiori e sulla disponiblità di un gran numero di librerie di supporto, che permettono di lavorare a più alto livello.

6.1.1 Marching Cubes

L'algoritmo marching cubes è utilizzato per costruire una rappresentazione poligonale di isosuperfici a partire da dati volumetrici [LC87]. Una isosuperficie è caratterizzata da un valore costante detto isovalore.

L'idea alla base dell'algoritmo è rappresentare un voxel di forma cubica tramite i valori dei pixel agli otto angoli del cubo. Se tra i pixel appartenenti a un voxel del volume, ne esiste almeno uno con valore minore dell'isovalore, e almeno uno con valore maggiore, allora il voxel contribuirà alla costruzione dell'isosuperficie. Individuando gli spigoli del voxel che intersecano la isosuperficie, è possibile creare superfici composte da triangoli, che dividono il voxel in regioni interne ed esterne alla isosuperficie. Connettendo le superfici calcolate in tutti i voxel attraversati dalla isosuperficie, si ottiene una rappresentazione, come mesh di triangoli, della isosuperficie stessa.

Poichè una rappresentazione tridimensionale delle superfici anatomiche risulta molto utile in ambito medico, questo algoritmo è usato spesso per processare dati medici. Permette di ricostruire facilmente l'oggetto tridimensionale rappresentante un organo, a partire da sezioni di volumi tipicamente acquisiti da tomografie computerizzate e risonanze magnetiche.

L'algoritmo

Prima di illustrare l'algoritmo marching cubes, sarà presentato l'algoritmo marching squares che usa lo stesso approccio in 2D.

Si immagini di avere una griglia bidimensionale composta da quadrati, dove ogni vertice di ogni quadrato ha un peso associato. Interpolando i valori dei pesi lungo i lati del quadrato è possibile definire un valore per ogni punto del perimetro.

Dato un valore di riferimento $\mathcal{R}$ l'algoritmo traccia dei segmenti che uniscono punti sui lati dei quadrati il cui valore interpolato è uguale a $\mathcal{R}$.

Si consideri per esempio la griglia in figura 6.1. Nella figura sono mostrati pesi assegnati ai vertici e una curva il cui valore di riferimento, costante, è posto a 5.

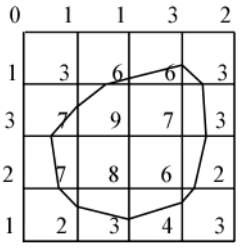


Figura 6.1: Esempio di griglia 2D per la ricostruzione di una linea

Il calcolo della curva dipende dal tipo di interpolazione usato : spesso è sufficiente l'interpolazione lineare. Il calcolo avviene considerando individualmente ogni quadrato della griglia. Tutte le possibili intersezioni che possono avvenire nel quadrato, sono rappresentate da 16 diverse configurazioni (vedi figura 6.2).

Estendendo l'algoritmo marching squares allo spazio tridimensionale si arriva alla definizione dell'algoritmo marching cubes. Nello spazio tridimensionale sono state enumerate 256 diverse configurazioni che, operando per rotazioni e simmetrie, possono essere generalizzate nelle 15 tipologie rappresentate in figura 6.3.

L'algoritmo analizza i voxel uno per volta, ricercando i voxel attraversati dalla isosuperficie da rappresentare. Si considera un voxel è come attra-

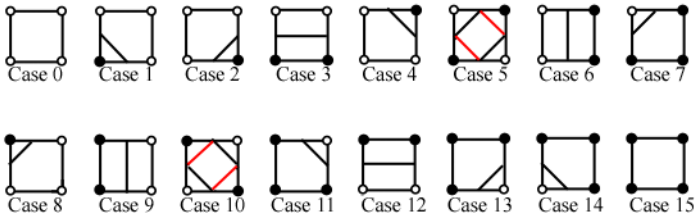


Figura 6.2: Le 16 configurazioni che rappresentano tutti i tipi di intersezioni possibili

versato dalla isosuperficie quando i valori interpolati sugli spigoli del cubo corrispondono all'isovalore della isosuperficie. Confrontando i punti di intersezione sui lati del voxel con i casi noti, si identificano le superfici interne al voxel.

Per analizzare il volume si parte dalla prima sezione, esaminando i voxel riga per riga. Quando una sezione viene completata, i punti della superficie costruita vengono considerati temporanei, fino al completamento della sezione successiva. Il confronto dei risultati del calcolo sulle due sezioni, fa sì che alcuni punti entrino a far parte della superficie e altri vengano scartati.

La composizione delle superfici calcolate per ogni sezione, costituisce la rappresentazione dell'isosuperficie. costituisce la rappresentazione della isosuperficie. In questo lavoro si è sfruttata un'implementazione dell'algoritmo marching cubes fornita dalla libreria VCG [MSS94] (vedi appendice B). In figura 6.4 è rappresentato un colon ricostruito con la tecnica descritta.

6.2 Rimozione delle strutture extra-colon

Come si è detto la rappresentazione 3D del colon è ottenuta applicando l'algoritmo marching cubes al volume che rappresenta l'addome del paziente. Il valore di riferimento usato per l'individuazione della isosuperficie è un valore di grigio che i voxel assumono al confine tra una zona piena d'aria e una zona occupata da tessuto. Pertanto l'algoritmo costruirà delle superfici in

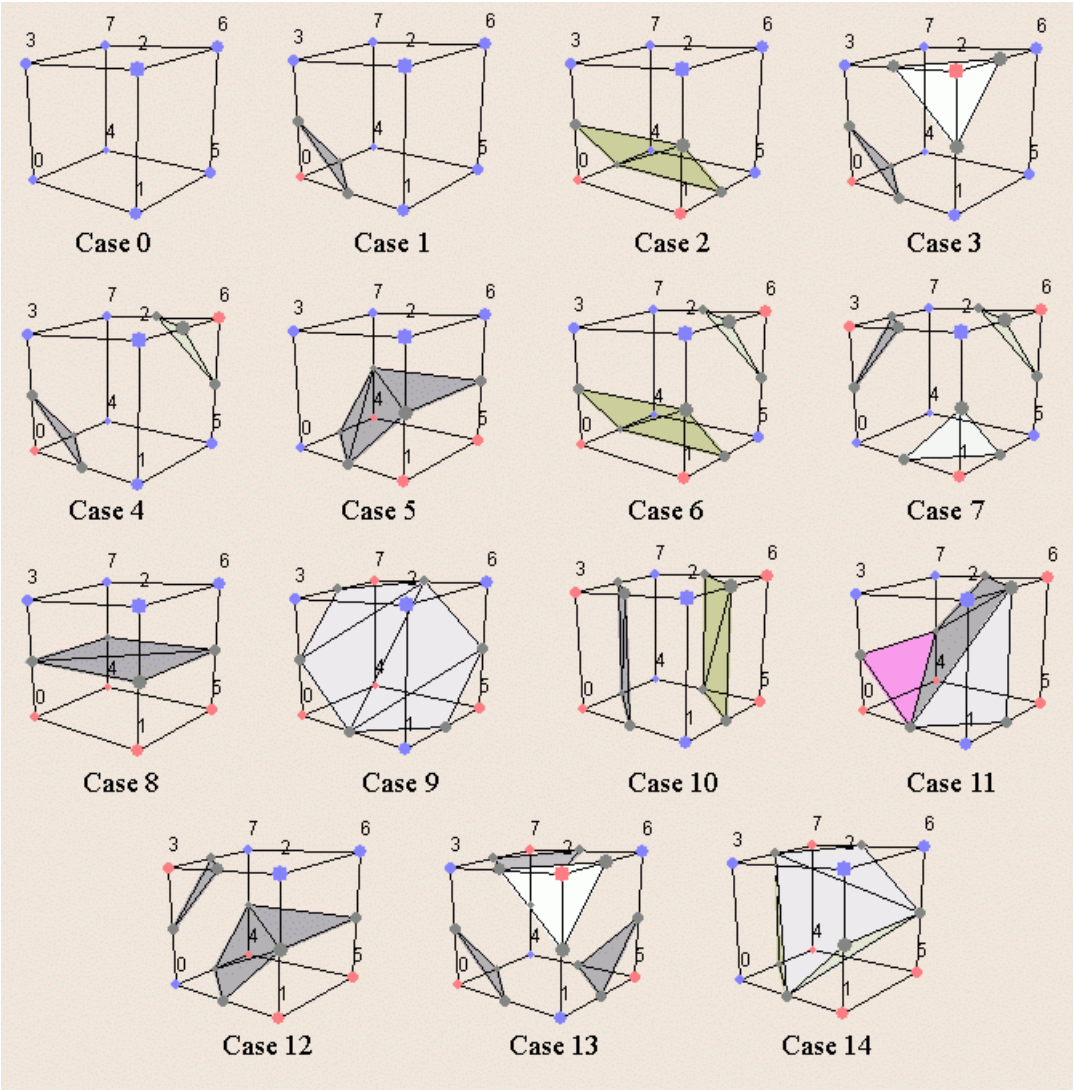


Figura 6.3: Le 15 tipologie di configurazioni che rappresentano tutti i tipi di linee possibili in uno spazio 2D

corrispondenza di voxel appartenenti al bordo delle strutture dell'addome che contenegono aria.

Tra le suddette strutture figurano il colon, i polmoni e in alcuni casi parti del piccolo intestino.

I polmoni risiedono tipicamente nelle prime 30-40 immagini che costituiscono il volume. L'algoritmo di rimozione dei polmoni, descritto nel pros-

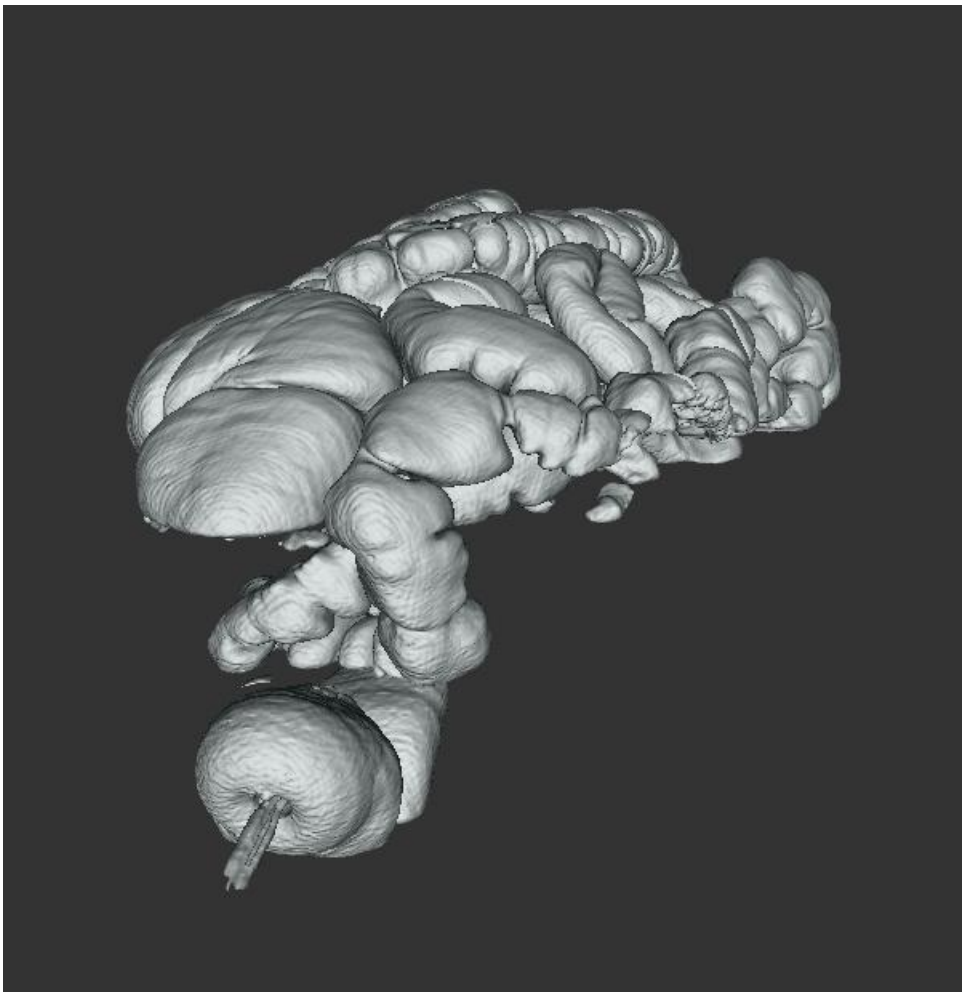


Figura 6.4: Un esempio di colon ricostruito con Marching Cubes e visualizzato come mesh di triangoli

simo listato, fa uso di questa caratteristica per individuare le mesh che li rappresentano e rimuoverle dal modello.

Sia $\mathcal{M}_{\mathrm{I}}$ la mesh costruita, l'algoritmo implementato è il seguente:

Algoritmo 6.2: Rimozione strutture polmoni e rumore

Input: Mesh di input $\mathcal{M}_{\mathrm{I}}$;

Output: Mesh di output $\mathcal{M}_{\mathrm{O}}$ ottenuta da $\mathcal{M}_{\mathrm{I}}$ rimuovendo le strutture corrispondenti ai polmoni;

1: $\mathcal{C} \leftarrow$ computeConnectedComponents($\mathcal{M}_{\mathrm{I}}$); // Calcola le componenti connesse
2: $\mathcal{B} \leftarrow$ computeBoundingBoxes($\mathcal{M}_{\mathrm{I}}, \mathcal{C}$); // Calcola i bounding box delle componenti connesse
3: $\mathcal{LB} \leftarrow$ findLungsBBox($\mathcal{B}$);
4: $\mathcal{M}_{\mathrm{O}} \leftarrow$ removeLungsAndInside($\mathcal{M}_{\mathrm{I}}, \mathcal{LB}$); // Rimuove i polmoni e le strutture in essi contenute

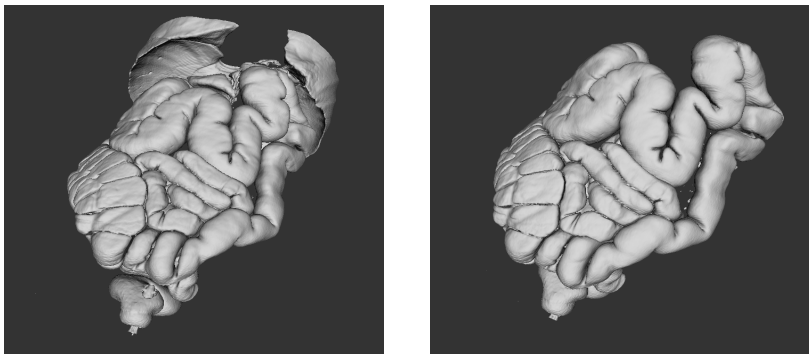
Nella riga 4 vengono tutte le componenti connesse della mesh. I polmoni, e le strutture minori in essi contenute, costituiscono componenti separate dal colon.

Nella riga 5 vengono calcolati i *bounding box* delle componenti individuate. Il bounding box di una mesh è il parallelepipedo più piccolo che ne contiene tutta la geometria.

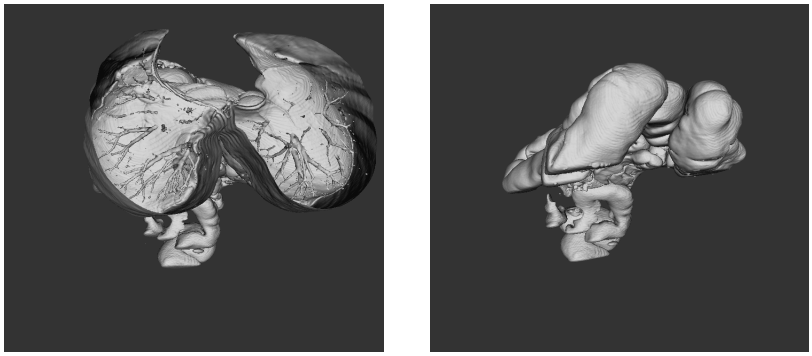
Nella riga 6 vengono individuati i bounding box dei polmoni come i due più grandi nel primo quarto del volume.

Nella riga 7 vengono rimosse tutte le componenti i cui bounding box coincidono con quelli dei polmoni, o sono in essi contenuti. Durante questa procedura vengono rimossi anche gli oggetti più piccoli, considerati di scarso interesse e associabli a rumore nell'immagine, costituiti da un numero di triangoli molto basso rispetto a quello della mesh completa.

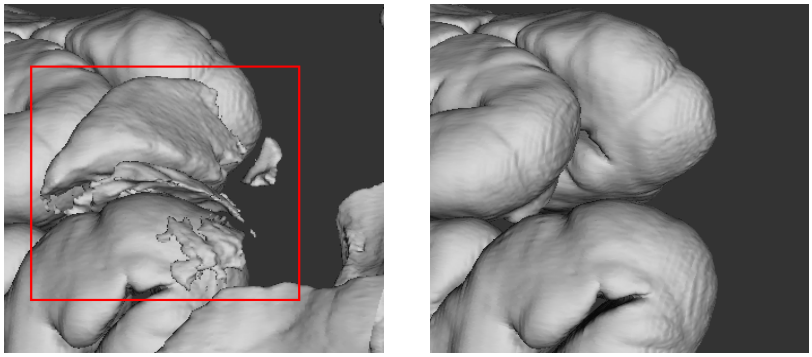
In figura 6.5 sono riportati esempi del risultato della rimozione dei polmoni e di altre strutture, nelle figure 6.6 e 6.7 due esempi di schermate dell'applicazione.



a) b)



c) d)



e) f)

Figura 6.5: a) e c) ricostruzione del colon in cui sono ancora presenti i polmoni; b) e d) dopo la rimozione dei polmoni; e) particolare che evidenzia piccole strutture esterne al colon e f) dopo la rimozione.

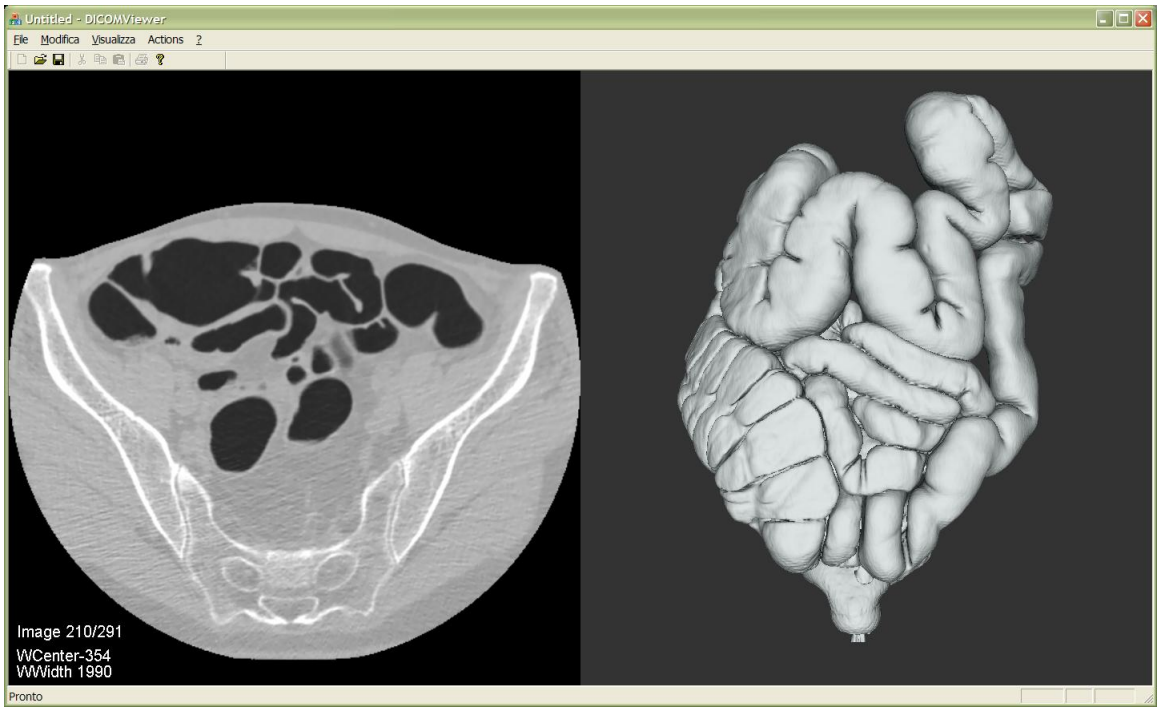


Figura 6.6: Un esempio di schermata dell'applicazione con affiancate le immagini e una vista esterna del colon

Figura 6.7: Un esempio di schermata dell'applicazione con affiancate le immagini e una vista interna del colon

Capitolo 7

Modellazione UML

La modellazione dei componenti del sistema deriva dalle linee guida di progettazione esposte nel capitolo 4. Di seguito sono discussi e presentati i risultati della modellazione. Saranno descritti i moduli che compongono il sistema. L'architettura è stata progettata con il linguaggio *UML*, il linguaggio standard per la modellazione di componenti software.

7.1 Sottosistemi e suddivisione delle funzionalità

Per rendere più semplice lo sviluppo di un sistema software complesso è sempre utile individuare gruppi di funzionalità diverse, che costituiscono blocchi indipendenti o con precise relazioni di dipendenza tra loro. È così possibile suddividere il sistema in sottosistemi, che facilitano sviluppo e manutenzione. Questo contribuisce inoltre a ridurre i tempi di compilazione, che in caso di sistemi di grosse dimensioni possono rallentare lo sviluppo. Di seguito sono elencati i sottosistemi individuati durante l'analisi del problema insieme a una breve descrizione delle funzionalità che offrono. In figura 7.1 sono mostrate le dipendenze tra i moduli.

DicomFileIO Fornisce le funzionalità per la lettura di file DICOM, per l'estrazione dei dati e per l'organizzazione dei file e del loro contenu-

to in dataset, raggruppandoli per paziente, studio, serie e immagine. Definisce inoltre le strutture dati di base per il trattamento dei dati.

Composer Raggruppa le funzionalità che permettono di comporre la finestra di disegno con View tra loro indipendenti, funzionalità per la gestione dell'input del mouse, del dragging, del resizing e di altre operazioni comuni su finestre. Fornisce anche un'astrazione per comandi e funzionalità utili per comporre l'interfaccia e il modello di interazione.

ImageProcessing Raggruppa un insieme di tool e funzionalità per il trattamento delle immagini, come ad esempio filtri e algoritmi di segmentazione. Viene usato sia per la visualizzazione delle immagini che in fase di elaborazione dei dati.

GLUtil Raggruppa funzioni di supporto al rendering con OpenGL, per esempio trasformare un'immagine in texture o disegnare caratteri. Vi si definiscono inoltre le View specializzate per OpenGL.

MeshUtil Raggruppa funzionalità di supporto per dati rappresentati come mesh di triangoli. Contiene ad esempio un'implementazione dell'algoritmo *Marching Cubes* che genera una mesh da una sequenza di immagini. Si appoggia alla libreria esterna *vcg*.

EndoscopySystem Rappresenta il cuore del sistema. Contiene le classi che permettono di assemblare i moduli del sistema. È qui che vengono definiti il modello di interazione e le modalità di visualizzazione.

Nelle sezioni successive verrà esposta l'asrchitettura di ogni modulo, descrivendo brevemente il funzionamento e le classi che lo compongono.

7.2 DicomFileIO

Un dataset in formato DICOM consiste in un insieme di file, eventualmente organizzati in directory, ciascuno dei quali contiene un'immagine e tutte le informazioni necessarie sia per estrarre i pixel che per collocare l'immagine all'interno del dataset. Quest'ultimo può infatti contenere esami di più pazienti, ciascuno dei quali ha associati uno o più studi, dove ciascuno studio

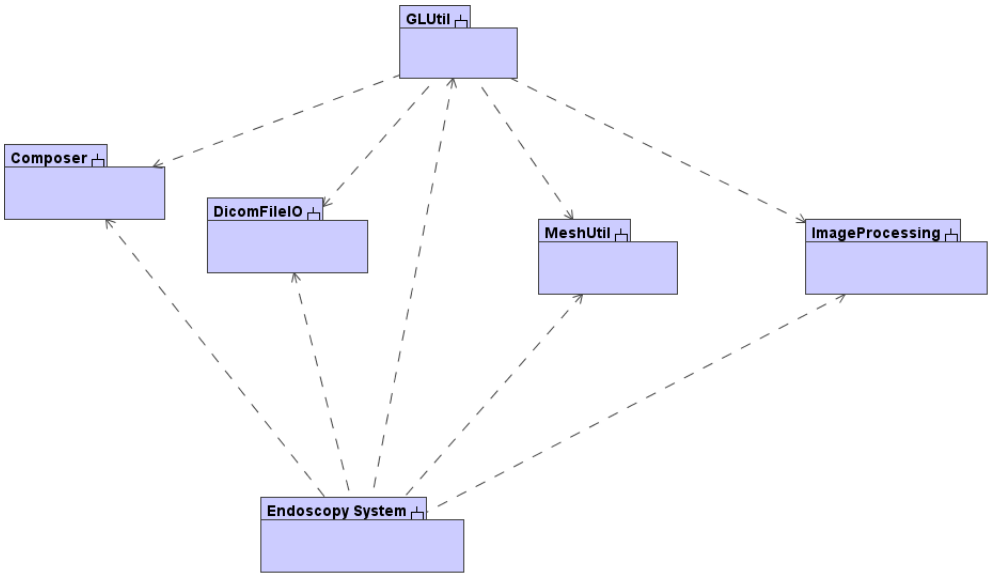


Figura 7.1: Dipendenze tra i moduli

può contenere una o più serie. Infine ogni serie contiene immagini (si veda appendice A).

La struttura del modulo riflette la struttura di un dataset DICOM (fig. 7.2). Le classi principali del sistema sono:

DicomPatient rappresenta un paziente. Contiene le informazioni per identificare il paziente e mantiene dei riferimenti agli studi associati;

DicomStudy rappresenta uno studio. Permette di identificare lo studio e mantiene riferimenti alle serie associate;

DicomSeries rappresenta una serie. Permette di identificare la serie e mantiene riferimenti alle immagini che la compongono;

DicomImageProxy funge da proxy verso le immagini, che vengono caricate su richiesta e decodificate mediante PixelDecoder. Ha il percorso del file e le informazioni per inizializzare la lookup table dei valori di grigio;

DicomInputStream è la classe che si occupa dell'effettiva lettura dal file, usando DataDictionary per la decodifica degli elementi e producen-

do dei DicomDataElement che che vengono usati da DicomFile per registrare le informazioni;

DicomFile estrae le informazioni da un file avvalendosi delle funzionalità di DicomInputStream, le registra come DicomDataElement e costruisce il PixelDecoder per leggere decodificare l'immagine contenuta nel file;

DataDictionary è il dizionario dei dati. Definisce quali sono gli elementi individuabili in un file Dicom e per ciascun elemento fornisce il tipo e la dimensione. Ogni elemento, una volta letto, viene poi incapsulato in un DicomDataElement;

PixelDecoder estrae dal file i byte che costituiscono l'immagine e li decodifica fornendo i pixel utilizzabili per la visualizzazione dell'immagine. Estrae le informazioni per la decodifica dai DicomDataElement che contengono la specifica della modalità di registrazione dell'immagine su file. Usa le informazioni per costruire una maschera da applicare ai byte letti per calcolare il valore di un pixel;

DicomDataElement incapsula un elemento del file, individuato da una coppia (gruppo,elemento) e dal valore dell'elemento;

DicomDataset rappresenta il punto d'accesso al modulo. Si comporta come *Facade* (vedi sez. 4.2). Permette ai client del modulo di accedere alle funzionalità in modo semplice, rendendoli indipendenti dall'implementazione dal tipo delle classi che costituiscono il sottosistema DicomFileIO.

la classe DicomDataset ha lo scopo di semplificare l'uso del sottosistema, proponendosi come punto d'accesso per i client. Per questi ultimi sarà infatti sufficiente fornire alla classe la lista dei file che costituiscono il dataset. L'oggetto DicomDataset provvederà quindi a costruire l'albero che rappresenta il dataset e a fornire accesso alle serie di immagini. La classe DicomSeries fornisce effettivamente il secondo punto di ingresso al sistema, è infatti tramite tale classe che si possono scorrere le immagini, sfruttando riferimenti a oggetti di tipo DicomImageProxy che DicomSeries restituisce. DicomDataset, DicomSeries e DicomImageProxy sono le uniche classi di cui

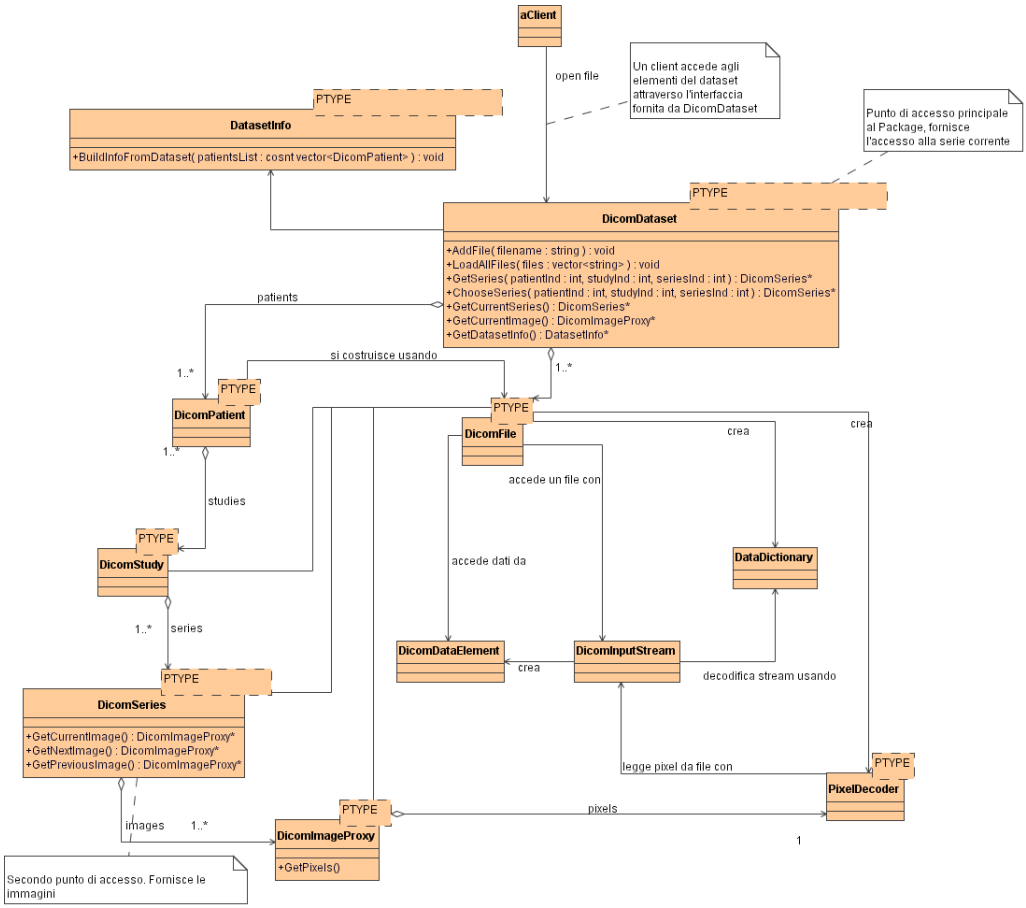


Figura 7.2: Organizzazione del modulo DicomFileIO

il client ha effettivamente bisogno per usare il modulo, poiché però le ultime due vengono comunque fornite dalla prima, DicomDataset costituisce l'unico vero punto di accesso al modulo.

DicomDataset eredita dal pattern Facade due vantaggi:

- la semplicità nell'uso del modulo;
- la riduzione dei tempi di compilazione, poiché quando si modificano solo le classi interne al modulo, non è necessario ricompilare i client.

7.3 Composer

Quando un'applicazione deve permettere all'utente un'interazione complessa con elementi e algoritmi dell'applicazione stessa, necessita di un'interfaccia che renda tali interazioni il più possibile semplici e intuitive. Nella maggior parte dei casi per lo sviluppo ci si avvale di librerie specifiche, che permettono di sviluppare una *Graphical User Interface* (GUI) adatta allo scopo.

Solitamente tali librerie gestiscono anche le finestre di disegno dell'applicazione, permettendo di suddividere la finestra principale in sottofinestre, di ridimensionare le finestre etc. Se però l'applicazione diventa troppo strettamente legata a tali librerie si possono creare problemi di portabilità e di riuso del codice. Cambiare libreria può infatti costringere a riscrivere parte del codice, che magari avrebbe invece potuto far parte dell'applicazione stessa. Talvolta può essere infatti necessario dover abbandonare una libreria, magari perché non più supportata, o perché è necessario riconfigurare l'interfaccia per un altro sistema operativo per il quale la libreria a cui ci si era affidati non è disponibile.

Il desiderio di rendere il nostro sistema il più indipendente possibile da librerie esterne, insieme alla necessità di avere un'interfaccia estremamente flessibile e personalizzabile anche a run time, ha portato all'introduzione di uno strato che si frapponesse tra il cuore del sistema e la libreria dedicata alla GUI.

Il modulo *Composer* costituisce tale strato e ricopre i seguenti compiti:

- creazione delle componenti che costituiscono le finestre;
- ridimensionamento delle finestre;
- gesione dell'input del mouse;
- disegno dei contenuti.

Non si occupa invece di:

- creazione di elementi tipici delle GUI, come pulsanti o scroll bar;
- creazione della finestra principale dell'applicazione, che conterrà gli elementi di Composer;

- funzionalità legate alla piattaforma, come ad esempio la creazione di un *Rendering Context* per il disegno basato su OpenGL.

ma fornisce strumenti per ridirigere gli eventi provenienti dalla GUI sulle funzionalità del sistema.

Gli elementi cardine del modulo Composer sono le classi:

- *WindowComposer*;
- *ComposerPanel*;
- *ComposerView*.

In figura 7.3 è rappresentata l'architettura del modulo.

Il sistema a finestre definito dal modulo Composer è basato su una struttura ad albero, nel quale WindowComposer rappresenta la radice e ComposerView (o le classi da essa derivate) le foglie. Descriveremo ora il funzionamento di ciascuna delle tre classi.

WindowComposer

La finestra principale dell'applicazione deve conoscere unicamente l'istanza di WindowComposer che rappresenta l'interfaccia in uso al momento. Nulla impedisce che vi siano più istanze di WindowComposer, che danno vita a interfaccie alternative, e che possono essere sostituite dinamicamente. Un'istanza di WindowComposer mantiene una lista di ComposerPanel, ognuno dei quali possiede una ComposerView.

WindowComposer ha essenzialmente il compito di instradare, nello strato successivo dell'albero, eventi dei tipi seguenti:

- richieste di disegno;
- ridimensionamento delle finestre;
- eventi del mouse.

Deve inoltre occuparsi della composizione della finestra, appoggiandosi alle funzionalità di ComposerPanel, a seconda delle richieste del client del modulo. Consente infatti di aggiungere pannelli contenenti ComposerView, le cui proporzioni e posizioni vengono specificate nei metodi associati.

ComposerPanel

La classe ComposerPanel rappresenta la una porzione della finestra gestita dal WindowComposer al quale appartiene e ha un riferimento verso la ComposerView che effettua il disegno nell'aera gestita. La classe

- gestisce lo *splitting*, permettendo di suddividere in maniera arbitraria la finestra in rettangoli di visualizzazione indipendenti, ciascuno rappresentato da un ComposerPanel e dalla ComposerView associata;
- gestisce in maniera congiunta con WindowComposer il ridimensionamento delle componenti della finestra;

ComposerView

ComposerView rappresenta l'elemento più complesso del modulo Composer. Tra i suoi compiti rientrano infatti:

- il disegno nella porzione di schermo assegnata;
- l'effettiva gestione dell'input, giunto attraverso i nodi superiori dell'albero;
- l'esposizione agli strati superiori dell'applicazione delle funzionalità necessarie per un *rendering* interattivo e per una parte dell'interazione con i dati.

Ciascuno dei tre compiti richiede che ComposerView sia realizzata in modo sufficientemente flessibile e generale da permettere tipi diversi di disegno e diverse modalità di interazione.

L'operazione di disegno riguarda diversi tipi di dati e si realizza nella visualizzazione di

- immagini (TAC, RM etc..) che costituiscono i dati;
- diversi tipi di rappresentazioni tridimensionali dei dati stessi (ad esempio mesh di triangoli);
- eventuali risultati nati dall'elaborazione dei dati.

Ciascuno dei suddetti tipi di visualizzazione potrebbe richiedere un tipo di interazione diverso.

Per rendere il disegno e l'interazione estendibili e modulari, si è ricorso al design pattern Decorator, come descritto nella sezione 4.2. Inoltre, poiché le operazioni di disegno sono basate sulla libreria OpenGL, parte delle funzionalità descritte nel capitolo 4 sono state modellate nel modulo GLUtil, e saranno quindi descritte nella sezione 7.5.

Le operazioni di disegno sono implementate mediante una catena di operazioni la cui parte terminale, l'oggetto da disegnare, è modellato con la classe GraphicsObj. Il pattern Decorator viene applicato in due modi, strettamente correlati:

- per aggiungere operazioni di disegno;
- per aggiungere gestori di interazione.

Le classi che aggiungono operazioni di disegno derivano da Decoration. Le classi che aggiungono gestori di interazione derivano da ViewFunctionality. ComposerView consente di aggiungere elementi di disegno e di interazione dinamicamente, espone inoltre la lista dei gestori di interazione e un meccanismo per disattivarli o attivarli.

Di seguito elenchiamo le altre classi del modulo, che concorrono all'implementazione dei meccanismi descritti.

GraphicsObject classe base per gli oggetti disegnabili, fornisce l'interfaccia comune a questo tipo di oggetti;

Decoration classe base per gli oggetti che aggiungono operazioni di disegno, modellati secondo il pattern Decorator;

ViewFunctionality classe base per i gestori di interazione, modellati secondo il pattern Decorator;

CallbackCommand classe che modella il pattern Command, permette di associare all'oggetto un comando e un ricevente. Tramite il metodo *Execute* permette di eseguire il comando. È implementata tramite la classe *function* della libreria boost (si veda l'appendice B).

7.4 ImageProcessing

Il modulo ImageProcessing raccoglie le classi che modellano gli algoritmi e le funzionalità descritti nel capitolo 5. Le classi principali che compongono il modulo sono:

ImageHist calcola l'istogramma di un'immagine;

ImageFilter contiene algoritmi per l'applicazione di filtri alle immagini, come i filtri media e anisotropo;

MathMorphology contiene algoritmi di morfologia matematica applicata alle immagini. Definisce ad esempio algoritmi per la rimozione del background di immagini TAC e per l'erosione e la dilatazione dell'immagine;

RegionGrowing implementa un algoritmo di region growing;

GrayLevelLUT permette di modificare dinamicamente e interattivamente la finestra dei toni di grigio visualizzabili per un'immagine;

Vi sono poi alcune classi di supporto all'implementazione degli algoritmi, come ad esempio una classe per estrarre facilmente i diversi tipi di *neighborhood* di un pixel, o una classe che contiente funzioni di utilità generale come la somma, la sottrazione, l'AND e l'OR tra immagini.

7.5 GLUtil

Il modulo GLUtil raggruppa le funzionalità di visualizzazione ed interazione specifiche per OpenGL e concorre con il modulo Composer alla modellazione delle funzionalità e dei requisiti descritti nel capitolo 4.

In figura 7.4 è rappresentata la struttura del modulo.

Le classi principali sono elencate di seguito:

GLOrthoObj modella il volume di vista con trasformazione di proiezione ortogonale;

GLPerspectiveObj modella il volume di vista con trasformazione di proiezione prospettica;

GLFont modella il disegno di caratteri con OpenGL;

FontOverlay deriva da Decoration e rappresenta un elemento della catena di disegno; ha un riferimento a un oggetto di tipo GLFont e fornisce la propria implementazione basandosi su questo;

GLTrackball permette di ruotare oggetti tridimensionali come se si agisse su una sfera che li contiene; deriva da Decoration e agisce come *wrapper* per la classe corrispondente della libreria vcg (vedi cap. B);

GLTrackBallInteraction modella l'interazione con GLTrackball; deriva da ViewFunctionality del modulo Composer;

OpenGLState modella lo stato di OpenGL, permette di controllare alcuni parametri;

LUTDecoration modella l'applicazione di una LookupTable all'immagine, deriva da Decoration; ha un riferimento verso un oggetto GrayLevelLUT del modulo ImageProcessing, mediante il quale fornisce la propria implementazione;

LUTInteraction modella l'interazione con LUTDecoration; deriva da ViewFunctionality;

VWorldPosition modella la posizione nel volume come posizione in un mondo virtuale; come classe astratta fornisce l'interfaccia per le classi specifiche;

ImageTrackerPos modella la posizione nel volume delle immagini, deriva da VWorldPosition;

CameraPos modella la posizione nello spazio delle rappresentazioni 3D, deriva da VWorldPosition;

GLGraphicsObject modella un oggetto di disegno specifico per OpenGL; deriva da GraphicsObject del modulo composer estendolo con funzionalità specifiche;

TexImage deriva da GLGraphicsObject, in quanto oggetto visualizzabile implementa il metodo Draw. Incapsula il comportamento per disegnare un'immagine, mantiene un riferimento all'oggetto DicomImageProxy associato mediante il quale ottiene l'immagine. Mediante TextureBuilder genera una texture dai dati e la disegna su un rettangolo di dimensioni appropriate;

WrapMesh modella un wrapper per le mesh della libreria vcg; deriva da GLGraphicsObject. Derivando da GraphicsObject implementa il metodo Draw mediante il quale vengono disegnate le mesh.

GLView sottoclasse di ComposerView specifica per il disegno basato su OpenGL. Modellata secondo il pattern Strategy, richiede tre componenti che implementino rispettivamente il modello di proiezione, il tipo di posizionamento e l'oggetto da disegnare; ad esempio per la visualizzazione delle immagini può essere composta da un GLOrthoObj, un ImageTrackerPos e un TextureImage;

Come già accennato, le sottoclassi di Decoration e ViewFunctionality vengono usate per la creazione delle catene di visualizzazione e di reazione al'input.

7.6 MeshUtil

Contiene classi che fungono da wrapper per la libreria vcg. Le più importanti sono:

DCAMesh wrapper per le classi mesh, definisce il tipo dei vertici e delle faccie che compongono la mesh;

WrapVCGMarchingCubes wrapper per l'algoritmo Marching Cubes implemetato nella libreria, fornisce un punto di accesso facilitato che raggruppi le operazioni di setup;

7.7 EndoscopySystem

Fino a questo punto abbiamo descritto i moduli del sistema e come ciascuno di essi concorre alla creazione di un sistema per l'endoscopia virtuale.

Ora verrà descritto come gli elementi dei sottosistemi vengono assemblati e come sono modellate e implementate le comunicazioni fra gli oggetti. In figura 7.5 è rappresentata l'architettura del modulo responsabile dell'assemblamento dei sottosistemi.

Le classi principali del modulo sono tre : *ExtendedDataset*, *DatasetDirector* e *EndoscopySystem*. Di seguito verrano elencati compiti e caratteristiche delle suddette classi, infine verrà descritto il modello di cooperazione tra di esse ed esemplificata la creazione di un sistema di colonscopia virtuale.

Vediamo quindi le classi :

ExtendedDataset deriva dalla classe DicomDataset del modulo Composer. Rappresenta un'estensione del dataset DICOM. Ai dati dell'esame aggiunge infatti una rappresentazione tridimensionale costituita da mesh di triangoli. Fornisce anche alcune funzionalità di supporto. Ad esempio trasforma una posizione, rappresentata dalle coordinate (x, y, z), dal sistema di riferimento delle immagini a quello delle mesh. È un template che richiede come parametro il tipo di generatore di mesh.

DatasetDirector è la classe centrale del sistema. Progettata seguendo il pattern *Mediator* definisce come avvengono le comunicazioni tra gli oggetti che costituiscono lo scheletro del sistema. Ha infatti riferimenti alle istanze di ComposerView che constituiscono l'interfaccia e all'ExtendedDataset. Incapsula le politiche di comunicazione tra gli oggetti dei suddetti tipi, permettendo di propagare la notifica di eventi e la circolazione dei dati ai soggetti coinvolti. Sfruttando il meccanismo dei Command descritto nel modulo Composer, permette di esporre funzionalità e comandi ai client.

EndoscopySystem ha il compito di assemblare le parti del sistema, per il quale fornisce un punto d'accesso in modo da semplificarne l'uso e sopratutto la creazione. Mantiene riferimenti verso le istanze di WindowComposer, ExtendedDataset e DatasetDirector attive. A un client

che usi un oggetto di tipo EndoscopySytem spetta il compito di creare i singoli componenti e di fornirli all'oggetto. La classe fornisce però un metodo virtuale *Build*, che rispecchia il pattern *template method*, il quale appoggiandosi ai metodi *BuildDataset*, *BuildDirector* e *BuildComposer* fornisce uno scheletro di procedura per costruire e configurare i componenti. Poiché però il tipo e la configurazione dei componenti sono strettamente legati alle caratteristiche del sistema che si vuole costruire, per i suddetti metodi non è fornita un'implementazione. Il client che necessita una metodologia di creazione può usare una specializzazione di EndoscopySystem che fornisca un'implementazione dei metodi, oppure può definirne una propria. La classe ColonographySystem è un esempio di specializzazione.

ColonographySystem Sottoclasse di EndoscopySystem, fornisce un'implementazione dei metodi per la creazione dei componenti. Configura gli oggetti WindowComposer e DatasetDirector. A quest'ultimo fornisce un ExtendedDataset configurato per gestire una rappresentazione 3D dei dati basata su mesh di triangoli e per generare le mesh mediante la classe WrapMarchingCubes. Costruisce istanze di specializzazzioni di ComposerView in modo che siano compatibili con il DatasetDirector scelto.

Vedremo ora in cosa consiste la configurazione del sistema prendendo come esempio ColonographySystem.

Nel metodo BuildDataset all'oggetto ExtendedDataset viene fornito un costruttore di mesh che a partire dalle immagini costruisca una rappresentazione tridimensionale degli elementi contenuti nelle immagini. L'oggetto viene costruito fornendogli come parametro di template la classe WrapMarchingCubes, decritta in MeshUtil.

Poiché per attuare le proprie politiche DatasetDirector deve distinguere tra le view a cui fa riferimento, deve conoscere il tipo di ciascuna di esse. Nel metodo BuildDirector vengono create due istanze specializzate di GLView, una per la visualizzazione delle immagini e una per la visualizzazione tridimensionale del colon. Le due view vengono poi fornite al director con i metodi appositi. BuildComposer costruisce il WindowComposer con le view

già settate al director. Il modo e l'ordine in cui vengono inserite nel composer determina la posizione che occupano su schermo.

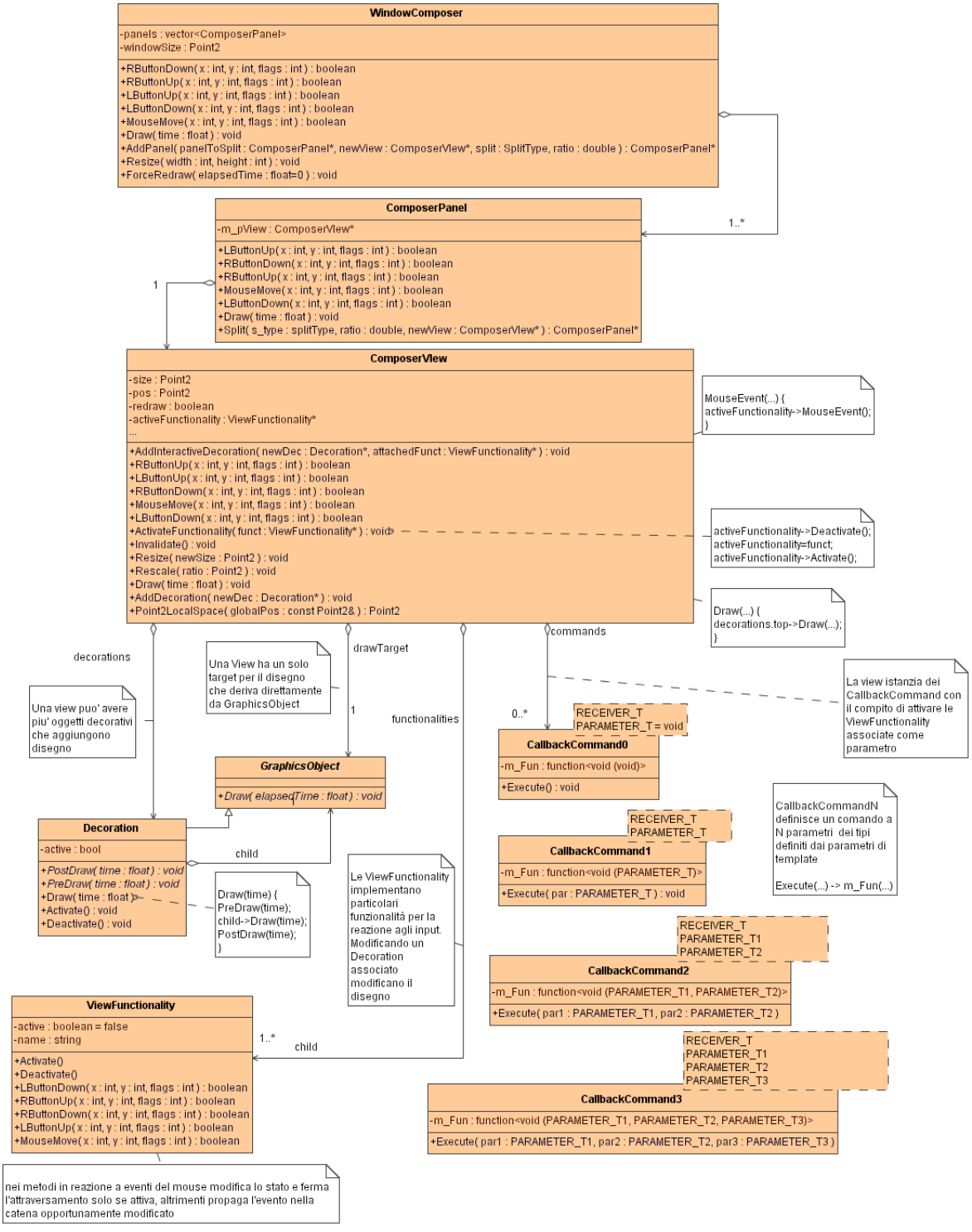


Figura 7.3: Schema del modulo Composer

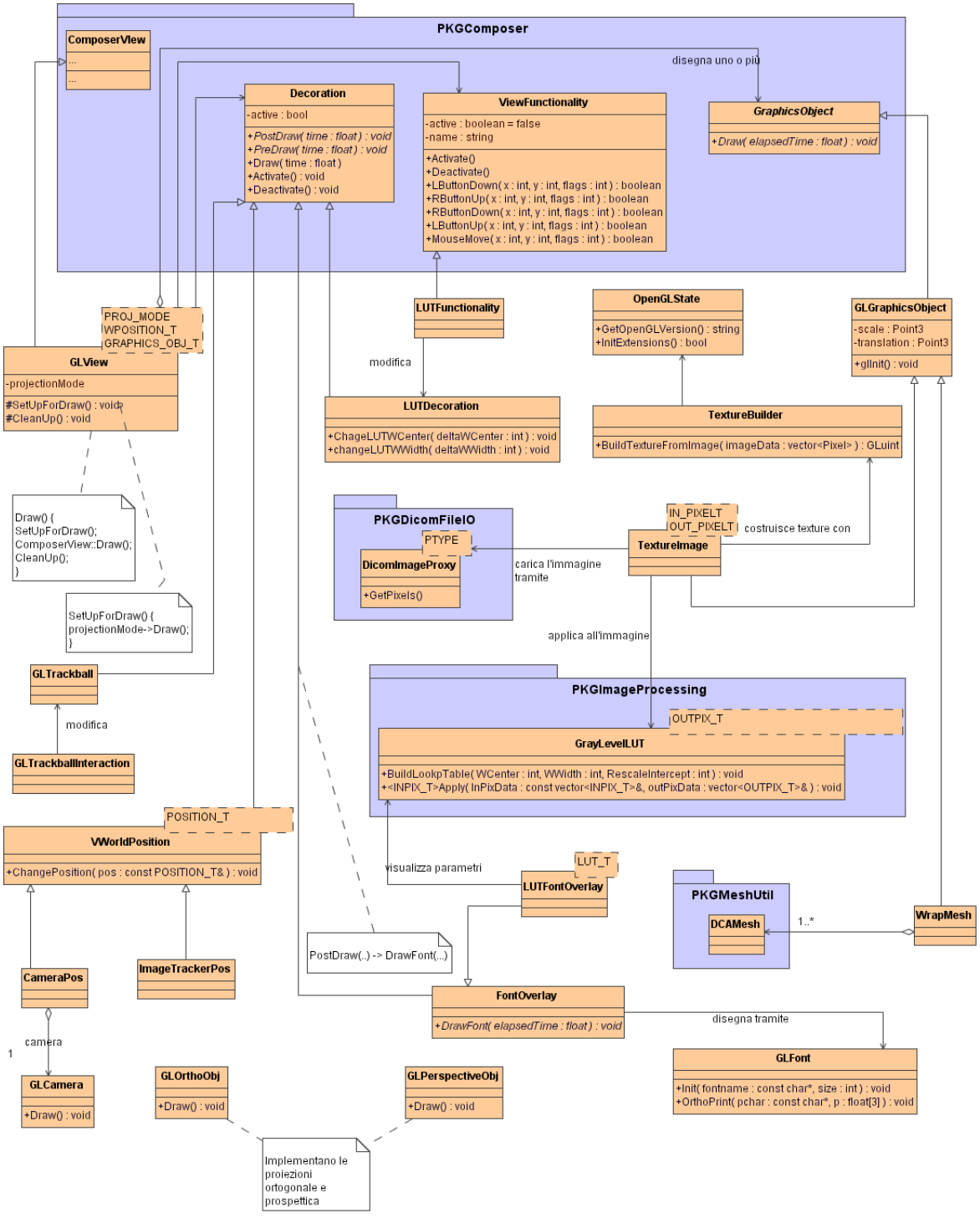


Figura 7.4: Architettura del modulo GLUtil

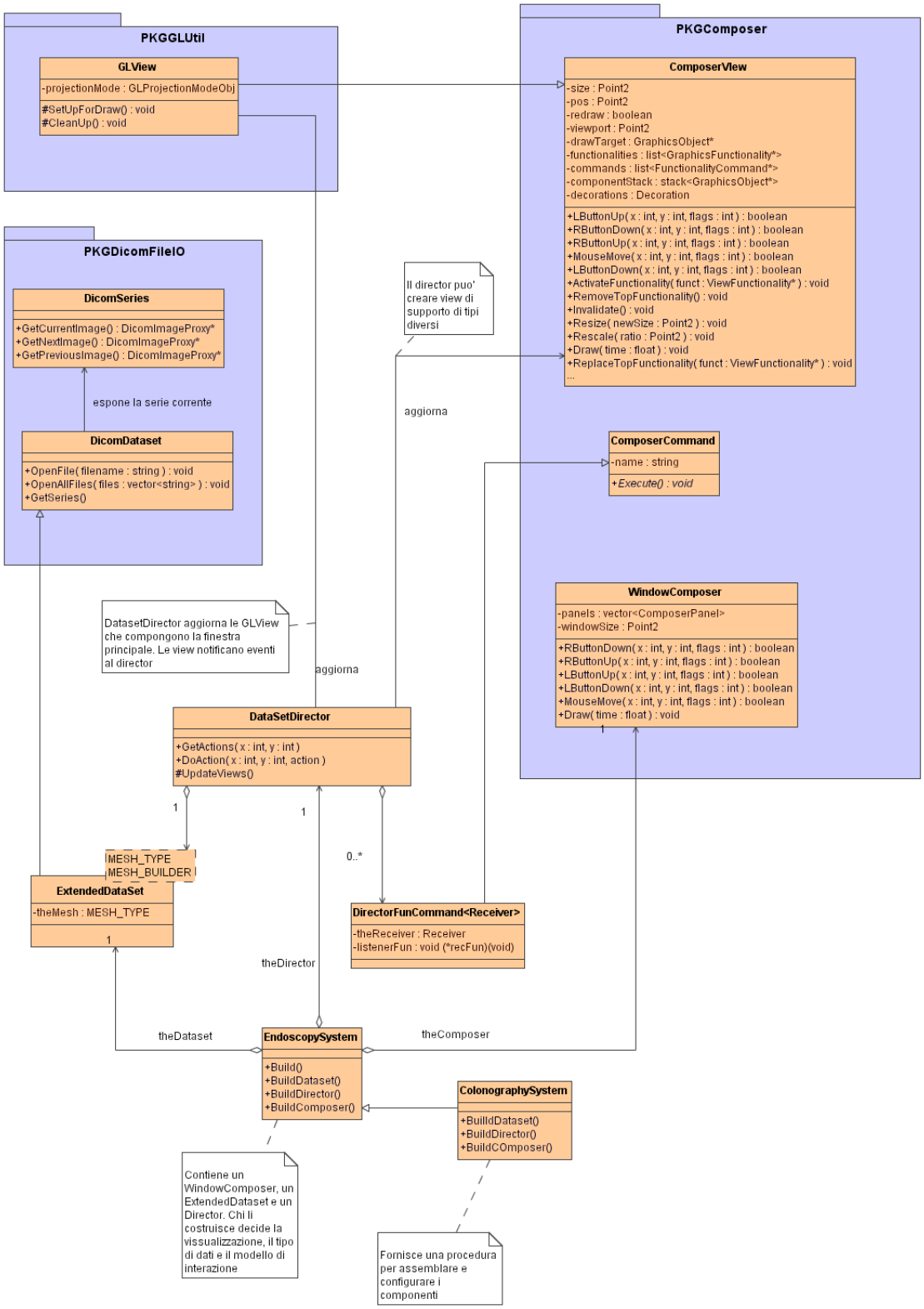


Figura 7.5: Architettura del modulo EndoscopySystem

Capitolo 8

Conclusioni e sviluppi futuri

Grazie ai progressi tecnologici degli ultimi anni, è sempre più diffuso in ambito medico l'uso di esami basati sull'analisi di immagini. Esami di questo tipo, come tomografie computerizzate e risonanze magnetiche, sono caratterizzati da un elevato numero di immagini che ne rende complicata l'interpretazione.

Da qui nasce l'esigenza di strumenti che offrano un supporto ai radiologi nella fase di diagnosi. Tali strumenti sono chiamati sistemi di computer assisted diagnosis o CAD.

Questa tesi è nata con l'obbiettivo di sviluppare una sistema CAD per colonscopie virtuali. L'esame di coloncopia virtuale permette una visita virtuale del colon, che facilita l'individuazione di lesioni, dette polipi, che costituiscono l'origine della maggior parte dei tumori al colon.

Ci si è resi conto che che gran parte dei requisiti di un sistema per la colonscopia virtuale, emersi dalle interviste con i radiologi, sono comuni a qualsiasi sistema CAD. Si è quindi deciso di progettare un sistema che potesse essere facilmente esteso per adattarsi a diverse tipologie di esami clinici basati su immagini.

Il risultato ottenuto è un framework flessibile ed estendibile, che costituisce una base sulla quale è possibile costruire applicazioni specializzate per esami diversi.

In particolare il sistema è stato specializzato per colonscopie virtuali. Affianca alla classica visualizzazione 2D delle immagini una ricostruzione tridimensionale del colon. Il sistema permette di interagire con i dati in

modo combinato, sia tramite la loro rappresentazione bidimensionale che tridimensionale. È possibile, per esempio, selezionando una regione nelle immagini, ottenere, nel modello tridimensionale, una vista della regione dall'interno del colon ed estrarne informazioni di tipo volumetrico per successive elaborazioni.

Possibili sviluppi futuri riguardano:

- integrazione di algoritmi di analisi per la ricerca automatizzata di polipi;
- specializzazione del framework per tipologie di esami diversi dalla colonscopia;
- esplorare in maniera più approfondita le possibilità di interazione offerte dalla combinazione di 2D e 3D.

Ringraziamenti

Ringrazio di cuore mio padre, mia madre, Gianni, Patrizia, Marcello e tutta la mia famiglia perché non mi hanno mai fatto mancare il loro appoggio, soprattutto nei momenti di difficoltà.

Ringrazio Franco Alberto e la professoressa Starita per l'aiuto costante e per la pazienza e disponiblità dimostrate durante tutto il lavoro.

Ringrazio Omar, Francesco, Michela, Manuela, Massimo e Irene, Marco e Chiara, Stefano e Silvia e tutti gli amici conosciuti a Pisa, per aver reso più lieti e leggeri questi anni di università.

Ringrazio Luca, Salvatore, Fabio, Federico, Valerio con tutti gli amici di Sassari, che hanno condiviso con me alcuni dei momenti più importanti della mia vita.

Ringrazio Daniele, Luca, Giuseppe, Fabio e gli altri ragazzi del laboratorio che mi hanno sopportato tutti questi mesi.

Un ringraziamento davvero speciale va a Letizia, per l'inestimabile supporto che mi ha fornito per tutto il tempo della tesi, ma soprattutto per aver reso questi anni insieme davvero speciali.

Appendice A

Lo Standard DICOM

Cenni storici

Alla fine degli anni'60, le nuove tecniche diagnostiche basate su immagini suscitarono l'interesse non solo della comunità medica, ma anche quello dei grandi produttori di dispositivi elettronici.

La maggior parte delle grandi aziende già presenti nel settore elettronico introdussero dispositivi per l'acquisizione, la memorizzazione e lo scambio di immagini, che, con il passare degli anni, divennero sempre più complessi e difficili da gestire.

Le difficoltà maggiori non derivavano comunque dalla complessità dei singoli dispositivi quanto dalla completa incompatibilità tra sistemi costruiti da case differenti. Tutti i produttori erano infatti d'accordo nel giudicare il nuovo mercato una sicura fonte di guadagno e cercavano, di conseguenza, di assumere una posizione monopolistica attraverso l'introduzione di protocolli e formati proprietari, rendendo di fatto estrememente difficoltoso qualsiasi processo di integrazione.

Lo stato delle cose divenne insostenibile quando crebbe la diffusione di dispositivi e centri di imaging : la proliferazione di architetture, dispositivi e sistemi operativi diversi rese estremamente complicata la gestione di questo tipo di apparecchiature.

L'American Institute of Physicists in Medicine (AAPM) cercò di disciplinare almeno la codifica delle immagini [BHM82], ma lo standard non

riuscì ad affermarsi a causa dell'opposizione dei grandi produttori, che non parteciparono alla sua stesura.

Nel 1983 l'American College of Radiology (ACR) e la National Electric Manufacturers Association (NEMA) avviarono una collaborazione volta alla definizione di un nuovo e più completo standard. La prima versione dello standard ACR-NEMA (ACR-NEMA 300-1985 1.0) venne presentata nel 1985. Nello standard erano comprese regole che disciplinavano sia gli aspetti hardware che software dei dispositivi [HPB+00, Pat99].

I vincoli imposti dallo standard, considerati troppo eccessivamente restrittivi da alcuni costruttori, uniti ad alcuni errori, costrinsero le due associazioni a riunirsi nuovamente per formulare una seconda versione, pubblicata nel 1988 (ACR-NEMA 300-1988 2.0).

ACR-NEMA 2.0 risultò obsoleto già al momento della sua introduzione in quanto non prevedeva alcuna specifica per l'interfacciamento dei dispositivi con reti di comunicazione. Questa carenza impediva una piena integrazione dei sistemi con sistemi informativi ospedalieri (HIS, Hospital Information System; RIS, Radiology Information System) e dei cosiddetti PACS, Picture Archiving and Communication Systems, sistemi di archiviazioni in grado di gestire un notevole numero di immagini digitali [LMBH90, Bak92].

Le prime due versioni dello standard, inoltre, poggiavano su un modello implicito dell'informazione usato nei dipartimenti di radiologia. La diffusione delle nuove tecniche anche in altri rami della medicina rendeva quindi necessaria sia un'estensione del modello per coprire le nuove realtà, sia una sua definizione più precisa dei termini usati, in modo da eliminare qualsiasi ambiguità nella sua interpretazione [HPB+00].

ACR e NEMA preferirono ripensare l'intero standard piuttosto che applicare piccole correzioni alla seconda versione. Dopo alcuni anni di lavoro presentarono, nel 1993, la terza versione, alla quale diedero il nome di DICOM 3, *Digital Imaging and Communication in Medicine*.

Organizzazione di DICOM 3

DICOM 3 è suddiviso in 12 parti (A.1), più vari documenti annessi (Annexes) [Pat99, Rev97]:

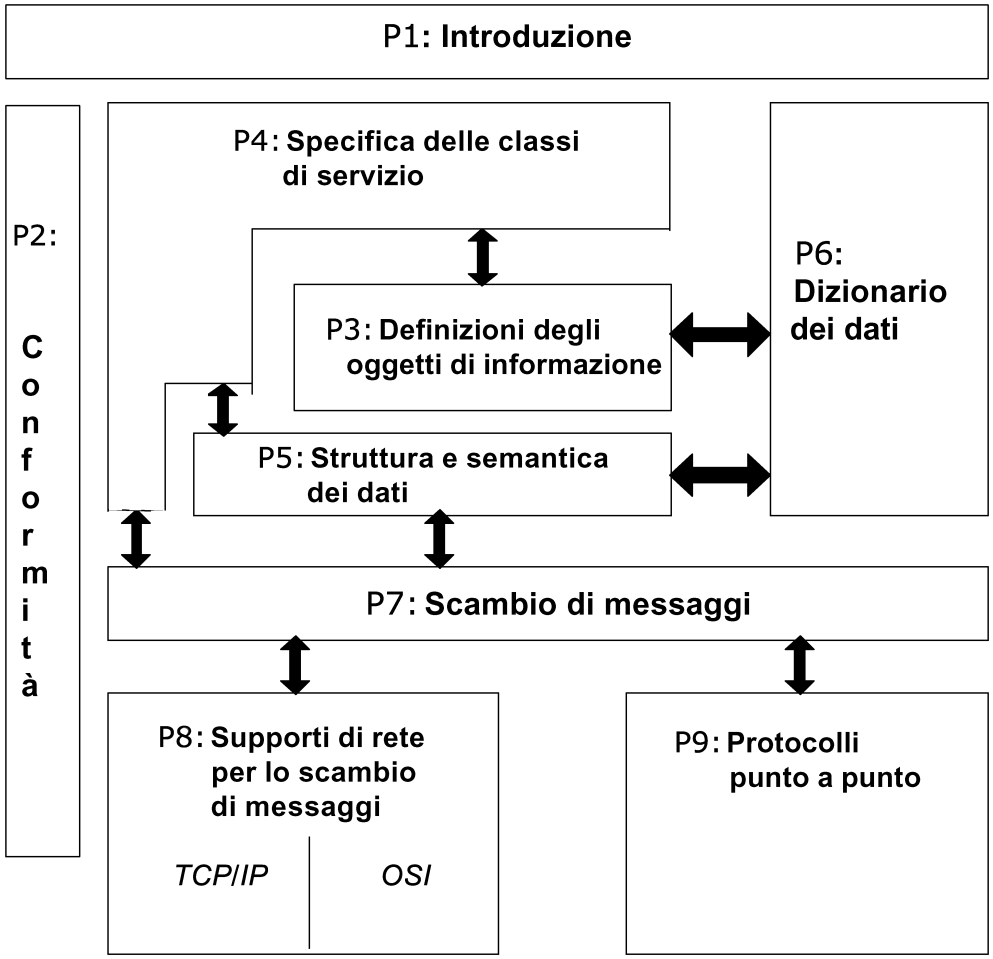


Figura A.1: Struttura di DICOM 3

Parte 1: Definizioni generali. Panoramica dello standard e descrizione dei principi basilari;

Parte 2: Definizione di conformità. Descrive le procedure che i costruttori devono seguire per essere legittimati a dichiarare le proprie apparecchiature conformi allo standard;

Parte 3: Oggetti di informazione. Definisce gli oggetti di informazione (IOD, Information Object Definition), astrazione di entità reali, il cui uso lo standard intende regolare;

Parte 4: Specifica delle classi di servizio. Definisce, per ogni IOD le operazioni legali su di esso;

Parte 5: Struttura e semantica dei dati. Specifica il formato in cui devono essere codificati i dati affinchè due sistemi DICOM possano dialogare e scambiarsi informazione;

Parte 6: Glossario dei dati. Elenco completo di tutti gli elementi di base che compongono gli IOD;

Parte 7: Messaggi di scambio. Definisce l'interazione tra software applicativi e sistemi DICOM;

Parte 8: Supporti di rete (TCP/IP, OSI). Protocolli di rete supportati. È previsto un supporto sia per LAN che per WAN;

Parte 9: Protocolli point-to-point. Supporto per i protocolli di rete usati nelle versioni precedenti dello standard;

Parte 10: Formato dei file. Regole per memorizzare uno IOD DICOM in un file e consentire uno scambio di informazioni off-line;

Parte 11: Applicazioni per sistemi di archiviazione. Definisce le modalità con cui le applicazioni gestiscono lo scambio informativo attraverso i supporti di memorizzazione;

Parte 12: Supporti fisici di memorizzazione. Specifiche per la scelta e la formattazione dei supporti fisici tramite i quali verranno scambiati dati DICOM.

Lo standard DICOM, a partire dalla sua prima versione, è stato leggermente modificato per correggere, ad esempio, problemi legati all'ambiguità e alla ridondanza di alcune definizioni. Negli ultimi anni il consorzio che segue la sua evoluzione cerca di ampliare il campo di applicazione di DICOM; ha, ad esempio, introdotto tre nuove parti per disciplinare la stampa, la visualizzazione e la sicurezza dei dati archiviati.

Modello dei dati

Spesso ci si riferisce allo standard DICOM come ad un modello *object-oriented*. Anche se usa termini come Information Object Definition, o Service Object Pairs (che possono ingannare) *non* è uno standard ad oggetti: uno studio più approfondito mostra come, nel suo sviluppo, sia stato seguito un classico approccio strutturato. I motivi risulteranno evidenti dopo la discussione del modello dei dati che segue.

Alla base dello standard si trova un modello del *mondo reale*, cioè della realtà rilevante per lo standard, che coincide quasi completamente con l'organizzazione dei dipartimenti di radiologia.

A partire da tale modello viene costruito il *DICOM Information Model*, diagramma Entità-Relazione, che modella gli oggetti del mondo reale e le relazioni in cui essi si trovano.

Una entità reale viene modellata con uno IOD, *Information Object Definition*, o oggetto informativo. Uno IOD rappresenta una classe di entità reali e può essere di due tipi:

- Normalizzato (*Normalized IOD*), quando contiene solo attributi inerenti alla entità che modella;

- Composto (*Composite IOD*), quando contiene, oltre ad attributi inerenti alla entità che modella, anche attributi semplicemente correlati;

Ad esempio, l'oggetto informativo "Paziente" è normalizzato in quanto contiene solo attributi inerenti ad un paziente (nome, età,...). L'oggetto informativo "Immagine di Risonanza Magnetica" è composto, in quanto, oltre ad attributi inerenti l'immagine, contiene attributi tipo "Nome del Paziente", che è solo legato all'immagine, ma non inerente.

Uno IOD è semplicemente una sequenza di attributi che rappresentano proprietà dell'entità modellata. Un attributo è definito come una coppia *(tag, valore)*, dove il *tag*, che individua univocamente l'attributo all'interno dello standard, è una coppia (gruppo, elemento). Una istanza di uno IOD viene creata, come è ovvio, fornendo un valore per tutti i tag obbligatori (alcuni tag sono opzionali o obbligatori sotto determinate condizioni).

Lo standard non modella solo le entità del mondo reale, ma anche le attività che vi hanno luogo. Le operazioni elementari vengono chiamate *Service Elements* (più precisamente, DICOM Message Service Elements, in quanto ogni interazione prevede l'invio di una richiesta e la ricezione di una notifica) e raggruppate in *Service Elements Group*. L'associazione tra IOD e operazioni legali viene effettuata attraverso SOP, cioè *Service Object Pairs*. Per ogni IOD possono essere definite più SOP, ad esempio, per lo IOD "Patient" esistono tre SOP distinte per le operazioni di "FIND", "MOVE" e "GET".

Risulta perciò evidente come l'approccio seguito nella progettazione dello standard, per quanto modulare, non sia ad oggetti, ma semplicemente strutturato: c'è, infatti, una netta distinzione tra dati ed operazioni. Le SOP potrebbero essere considerate oggetti, come fanno molti, ma in questo caso risulterebbe difficile spiegare la presenza di tre SOP distinte per l'oggetto "Patient" prima elencate.

DICOM è uno standard molto complesso, come d'altronde è ovvio, data la complessità della realtà cui può essere applicato. Nonostante sia oramai privo di qualsiasi ambiguità, l'implementazione di applicazioni ad esso conformi è un lavoro estremamente lungo. Le difficoltà derivano soprattutto dalla libertà che viene lasciata nella codifica dei dati: è vero che tale codifica è definita esattamente, ma è anche vero che, all'interno di DICOM, sono ammessi molti tipi di codifica ed interpretazioni. Un flusso di byte può essere codificato in little endian o big endian, la rappresentazione può essere in complemento a due o meno, ci sono due interpretazioni fotometriche dei pixel, diversi insiemi di caratteri supportati, i data element possono essere codificati con rappresentazione implicita del valore (gruppo, elemento, valore) oppure con rappresentazione esplicita (gruppo, elemento, tipo dell'elemento, valore). Da questo punto di vista, è auspicabile una semplificazione dello standard.

Immagini TC

Le immagini di Tomografia Computerizzata, in inglese Computed Tomography (CT), sono modellate con un *Composite IOD* (tabella A.1). Nella tabella è stata conservata la lingua inglese in modo che contenga termini non ambigui in quanto esattamente definiti da DICOM. Per *Information Entity* si

IE	Module	Usage
Patient	Patient	M
Study	General Study	M
	Patient Study	U
Series	General Series	M
Frame of Reference	Frame of Reference	M
Equipment	General Equipment	M
Image	General Image	M
	Image Plane	M
	Image Pixel	M
	Contrast/Bolus	C
	CT Image	M
	Overlay Plane	U
	VOI LUT	U
	SOP Common	M

Tabella A.1: Moduli dello IOD "CT Image"

intende il sottoinsieme di attributi di un Composite IOD correlato ad una singola entità del mondo reale; *Module* è definito come un sottoinsieme di attributi di una IE o di un Composite IOD logicamente correlati tra loro. La lettera nella colonna "Usage" significa:

- **M**: Mandatory, campo obbligatorio. Tutte le istanze dello IOD avranno tale campo non vuoto;
- **U**: User specified, opzionali;
- **C**: Conditional, obbligatori sotto alcune condizioni;

I moduli che hanno rivestito un ruolo particolarmente significativo per lo svolgimento del lavoro possono essere così raggruppati:

moduli per l'ordinamento dei dati che comprendono Patient ,General Study e General Series. Contengono attributi per identificare il paziente, lo studio e la serie a cui appartiene il file. Permettono così di raggruppare le informazioni secondo la struttura del dataset;

moduli per la decodifica dei dati volumetrici che comprendono General Image, Image Plane e Image Pixel. Consentono di estrarre e decodificare i pixel delle immagini e di ricostruire i dati volumetrici;

moduli per la presentazione dei dati che comprendono Modality LUT , e VOI LUT. Consentono di trasformare il valore dei pixel delle immagini in Hounsfield Units e permettono di scegliere la corretta modalità di presentazione dei voxel.

In figura A.2 sono rappresentate schematicamente le fasi di trasformazione che subiscono i voxel per la corretta presentazione.

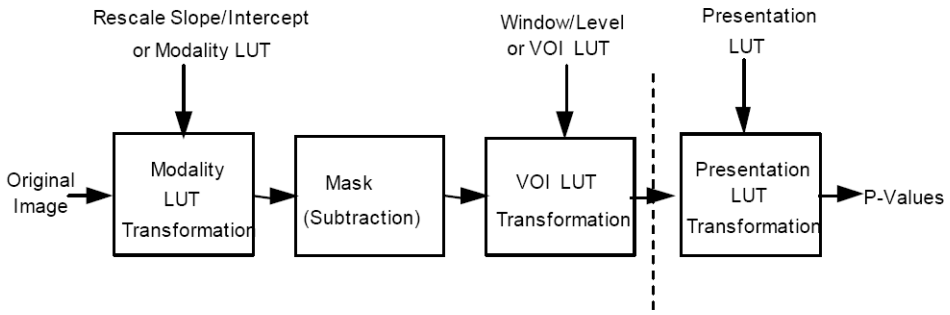


Figura A.2: Fasi di trasformazione delle immagini

Il passo riguardante Mask subtraction è applicabile solo in caso di immagini multiframe e non si applica alle immagini TC. In Presentation LUT possono essere applicate delle trasformazioni ulteriori, ad esempio rotazioni, ingrandimenti o shutter. Quest'ultima permette di rendere visibile su schermo solo una parte dell'immagine.

Per una descrizione dei restanti moduli si veda [ACR00, Parte 3 (2000), pag.32]

Appendice B

Librerie di supporto

OpenGL

OpenGL (*Open Graphics Library*) è una libreria per lo sviluppo di applicazioni grafiche interattive 2D e 3D. Può essere considerata una interfaccia software verso l'hardware grafico [OGL, WNDS01].

Il prefisso Open indica che OpenGL è una architettura aperta: il suo sviluppo è curato da una associazione di industrie leader del settore (ATI, Compaq, IBM, nVidia, Hewlett- Packard, Silicon Graphics) che hanno dato vita all'ARB (OpenGL Architecture Review Board). Questa organizzazione, nata nel 1992, ha il compito di sviluppare e standardizzare gli aggiornamenti di OpenGL, che oggi è giunta alla versione 2.0.

A partire dalla sua introduzione, OpenGL è diventata la libreria standard per lo sviluppo di applicazioni grafiche anche in ambito scientifico. Le applicazioni basate su OpenGL risultano facilmente portabili e producono risultati consistenti su ogni tipo di piattaforma compatibile, indipendentemente dal sistema operativo e dal sistema a finestre usato.

OpenGL mette a disposizione circa 250 primitive che comprendono funzioni per la manipolazione dei pixel, per la proiezione di poligoni in 3D e per la gestione del movimento degli stessi, per la rappresentazione di luci e colori, per il texture mapping e così via.

Le funzioni OpenGL sono relative all'utilizzo delle primitive fondamentali quali vertici, linee e poligoni piuttosto che a modelli complessi la cui gestio-

ne viene di norma lasciata nelle mani del programmatore, garantendo così la massima flessibilità. Naturalmente non mancano le possibilità di creare luci, trasparenze, riflessi e altri effetti speciali per trasformare una sterile rappresentazione di poligoni in una più fedele riproduzione della realtà.

OpenGL, nelle sue distribuzioni, è sempre accompagnata dalla libreria GLU (GL Utility Library). GLU è una libreria ausiliaria che contiene alcuni funzioni di più alto livello, utili ad esempio per la creazione di sfere o altri oggetti complessi ma anche per impostare particolari matrici di proiezione e per facilitare alcune operazioni al programma client. Normalmente, infatti, le funzioni della GLU contengono al loro interno routine basate sulle funzioni base di OpenGL.

Un'ulteriore libreria che è possibile integrare è la GLUT (GL Utility Toolkit), si tratta di un toolkit per la creazione delle finestre indipendente dal sistema operativo che permette, utilizzando solamente le funzioni di tale libreria, di scrivere programmi completi nascondendo le differenze di implementazione tra i gestori finestre dei vari sistemi operativi.

Boost

Boost è un insieme di librerie C++ mantenuto da una comunità di sviluppatori [Boo].

Il gruppo Boost è stato fondato da alcuni componenti del comitato per la standardizzazione del C++ e tuttora esiste una stretta relazione tra i due gruppi di lavoro. Circa due terzi delle nuove librerie aggiunte allo Standard C++ derivano direttamente da librerie proposte da Boost.

Una interessante caratteristica di Boost è il processo di integrazione di nuove librerie. Chiunque puo' proporre la propria libreria al vaglio del gruppo. Se questa soddisfa i requisiti minimi richiesti (per esempio, a garanzia di portabilità, deve poter essere compilata con almeno due compilatori diversi), inizia un periodo di rassegna ufficiale nella comunità Boost. In questo periodo sono valutati diversi aspetti della libreria come la progettazione, l'implementazione e l'accuratezza della documentazione. Alla fine del periodo di valutazione, un *review manager* decide se accetare o rifiutare la libreria.

Boost contiene decine di librerie che spaziano notevolmente per scopi e

dimensioni. In questa tesi si è usata la libreria *Lambda*. Questa libreria realizza una forma di lambda astrazione per il C++. Il suo nome deriva dalla programmazione funzionale e dal lambda-calcolo, dove una lambda astrazione rappresenta una funzione senza nome. La funzione principale della libreria Lambda è esporre un modo semplice per definire oggetti funzione senza nome per algoritmi STL.

Vcg

È una libreria che contine utility per le mesh e la visualizzazione con OpenGL. Contiene l'implementazione dell'algoritmo Marching Cubes usato per la ricostruzione 3D del colon.

Bibliografia

[ACR00] ACR/NEMA. *DICOM Standard Status (Base Standard 2000)*, http://www.dclunie.com/dicom-status/status.html 2000.
Documenti ufficiali, anche se continuamente nello stato *draft*, che descrivono lo standard DICOM.

[ACS] www.cancer.org. American Cancer Society.

[Bak92] A.R. Bakker. Communication between hospital and radiology information systems and picture archiving and communications systems. In M. Osteaux, editor, *Hospital integrated picture archiving and communication systems*, pages 55–97. Springer, Berlino, 1992.
Descrive le problematiche relative all'integrazione dei vari standard.

[BHM82] B.S. Baxter, L.E. Hitchner, and G.Q. Maguire. A standard format for digital image exchange. Technical Report 10, American Institute of Physicists in Medicine (AAPM), New York, 1982.
Il primo standard relativo alla gestione di immagini mediche.

[Boo] www.boost.org. Boost.

[GHJV05] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley, 2005.

[GKKJ92] G. Gerig, O. Kübler, R. Kikinis, and F.A. Jolesz. Nonlinear anisotropic filtering of mri data. *IEEE Transactions on Medical Imaging*, 11(2):221–232, June 1992.
Ottima introduzione ai filtri anisotropi.

[GTA+] S.B. Gokturk, C. Tomasi, B. Acar, D. Paik, C. Beaulieu, and S. Napel. A new 3-d volume processing method for polyp detection.

[GTA+01] Salih Burak Göktürk, Carlo Tomasi, Burak Acar, Christopher F. Beaulieu, David S. Paik, R. Brooke Jeffrey Jr., Judy Yee, and Sandy Napel. A statistical 3d pattern processing method for computer aided detection of polyps in ct colonography. *IEEE Trans. Med. Imaging*, 20(12):1251–1260, 2001.

[HPB+00] S.C. Horiil, F.W. Prior, W.D. Bidgoos, C. Parisot, and G. Claeys. *DICOM: An introduction to the standard*. ANALYSER Sales Ltd, 2000.
Panoramica sullo standard disponibile all'indirizzo http://www.dicomanalyzer.co.uk/html/introduction.htm.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, pages 163–169, 1987.

[LMBH90] B.A. Levine, S.K. Mun, H.R. Benson, and S.C. Horii. Assessment of the integration of a his/ris with a pacs. In *Proceedings of the 4th SPIE Conference on Medical Imaging*, volume 1093, 1990.

[Mat75] G. Matheron. *Random Sets and Integral Geometry*. John Wiley, 1975.

[Mey05] Scott Meyers. *Effective C++, 55 Specific Ways to Improve Your Programs and Design*. Addison Wesley, 2005.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In R. D. Bergeron and A. E. Kaufman, editors, *Visualization '94 Proceedings*, pages 281–287, Washington D. C., USA, 1994. IEEE Computer Society, IEEE Computer Society Press.

[Nor90] N. Nordström. Biased anisotropic diffusion - a unified regularization approach to edge detection. *Image and Vision Computing*, 8(4):318–327, 1990.

[OGL] www.opengl.org. Open Graphics Library, libreria per lo sviluppo di applicazioni grafiche interattive 2D e 3D.

[Pat99] M. Paterni. Lo scambio delle immagini in medicina: Dicom 3.0. *Computer Programming*, 9(79):33–37, April 1999.
Ottima introduzione in lingua italiana allo standard DICOM.

[PBR+04] David S. Paik, Christopher F. Beaulieu, Geoffrey D. Rubin, Burak Acar, R. Brooke Jeffrey Jr., Judy Yee, Joyoni Dey, and Sandy Napel. Surface normal overlap: a computer-aided detection algorithm with application to colonic polyps and lung nodules in helical ct. *IEEE Trans. Med. Imaging*, 23(6):661–675, 2004.

[PM87] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. In *Proceedings of IEEE Workshop on Computer Vision*, pages 16–22, Miami, USA, November 1987.
Primo lavoro sulla diffusione anisotropa.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. 12(7):629–639, July 1990.
Primo lavoro sulla diffusione anisotropa.

[Rev97] B. Revet. *DICOM Cook Book*. Philips, 14January 1997.
Ottima introduzione allo standard DICOM scritta, per uso interno, dalla Philips.

[Roe96] J.B.T.M. Roerdink. Computer vision and mathematical morphology. In W. Kropatsch, R. Klette, and F. Solina, editors, *Theoretical Foundations of Computer Vision*, volume 11, pages 131–148. 1996.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, UK, 1982.

[Ser87] J. Serra. Morphological optics. *Journal of Microscopy*, 145(1):1–22, 1987.

[TI] www.tumori.net. Tumori in Italia: progetto di ricerca nato dalla collaborazione tra Istituto Nazionale Tumori e Istituto Superiore di Sanità.

[Vin95] L. Vincent. Lecture notes on granulometries, segmentation and morphological algorithms. In K. Wojciechowski, editor, *Proceedings of the Summer School on Morphological Image and Signal Processing*, pages 119–216. September 1995.

[WNDS01] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide*. Addison Wesley, third edition, 2001.

[YN01] Hiroyuki Yoshida and Janne Nappi. Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps. *IEEE Trans. Med. Imaging*, 20(12):1261–1274, 2001.

[YNM+01] H. Yoshida, J. Nappi, P. MacEneaney, D.T. Rubin, and A.H. Dachman. Computer-aided diagnosis scheme for detection of polyps at ct colonography. *RadioGraphics*, 2001.