Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Una Applicazione per la Visualizzazione Grafica e la Ricostruzione 3D di Immagini DICOM

I. Marra – R. Del Gaudio – C. Vanzanella

RT-ICAR-NA-06-10 **03-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Una Applicazione per la Visualizzazione Grafica e la Ricostruzione 3D di immagini DICOM

I. Marra[1] – R. Del Gaudio[1] – C. Vanzanella[1]

Rapporto Tecnico N.: **RT-ICAR-NA-06-10**

Data: **03-2006**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli

Una Applicazione per la Visualizzazione Grafica e la Ricostruzione 3D di Immagini DICOM

R. Del Gaudio[*], I. Marra[*], Carmen Vanzanella[*]

Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino, 111, 80131 - Napoli

Sommario

Negli ultimi decenni sono stati sviluppati diversi tipi di sistemi medicali non invasivi, atti all'acquisizione di immagini tridimensionali degli organi interni di persone viventi. L'interpretazione di tali immagini, ha posto le basi per la nascita di una tematica scientifica che in terminologia anglosassone viene definita *"medical imaging"* ed in italiano con una piu' precisa connotazione "diagnostica per immagini".
In questo lavoro, viene presentata una applicazione per la visualizzazione interattiva di immagini medicali, acquisite nel formato DICOM, e per la ricostruzione, a partire dalle stesse, del modello tridimensionale.
Il componente software, realizzato grazie all'ausilio delle librerie *Insight Toolkit (ITK), Visualization Toolkit (VTK) e Fast Light Toolkit (FLTK)*, tutte Open Source e Cross Platform, consente oltre alla visualizzazione delle *slice*, l'elaborazione delle stesse, nonche' la ricostruzione 3D, per una completa visualizzazione dei tessuti acquisiti, oltre ad una serie di controlli necessari per la navigazione nello spazio 3D.

1 Introduzione

La medicina fruisce costantemente degli strumenti che le scienze e la tecnologia mettono a disposizione per una sempre piu' completa comprensione dei fenomeni fisici, chimici e biologici che avvengono nel complesso sistema che e' l'organismo umano. Negli ultimi decenni alle varie discipline scientifiche sopra menzionate, si e' affiancata, in maniera trasversale l'informatica.

L'enorme potenzialita' di calcolo, la possibilita' di modellizzazione di fenomeni complessi, la possibilita' di immagazzinare e reperire con grande rapidita' informazioni e quella di rappresentare in maniera visiva realta' complesse, sono solo alcuni degli esempi piu'

appariscenti del contributo enorme che l'informatica fornisce alle varie tematiche delle scienze. In particolare, con questo lavoro ci si propone di fornire uno strumento informatico a supporto di una delle branche piu' importanti della medicina che e' la "diagnostica per immagini". Iniziata poco piu' di un secolo fa con la scoperta dei raggi X da parte del fisico tedesco Roentgen, ha avuto negli ultimi decenni un forte impulso, dovuto alle nuove metodologie di impiego degli stessi raggi X, che hanno portato alla Tomografia Assiale Computerizzata, all'impiego di ultrasuoni(ecografia), di radioisotopi (scintigrafia, PET), alla scoperta dei fenomeni di risonanza magnetica ecc. In tutti questi casi si pone in primo luogo il problema di fornire all'utente finale, sia esso medico o paramedico, uno strumento di impiego sufficientemente agevole e sicuro, per utilizzare appieno queste grandi potenzialita' diagnostiche messe a disposizione dall'evoluzione scientifica e tecnologica. Infatti, la visualizzazione impegna il piu' importante apparato sensoriale umano che e' quello della vista e di conseguenza il relativo processo interpretativo della mente umana che costituisce un mezzo semplice ed efficace per comunicare informazioni complesse e ricche di contenuti, ma il presupposto fondamentale e' che i dati raccolti dai sensori che analizzano i fenomeni non vengano alterati durante la fase di elaborazione che pero' deve consentire all'operatore la piµu ampia disponibilita' di mezzi atti ad esaltare la comprensione dei fenomeni. Di qui nasce l'esigenza di consentire all'utente finale una serie di operazioni di analisi, sintesi, esaltazione dei contorni, scelta di toni di grigi, roto traslazione etc. delle immagini fornite dall'apparato diagnostico. In secondo luogo, ma non per importanza, si tratta di fornire agli informatici che devono sviluppare strumenti di diagnostica per immagini, un insieme di strumenti, da cui il termine toolkit, capaci di consentire lo sviluppo di software applicativi in grado di seguire la continua evoluzione sia delle metodologie diagnostiche che quelle di rappresentazione delle immagini che devono essere interpretate. Si tratta, percio', di individuare gli strumenti che, partendo dalle immagini fornite dai mezzi diagnostici sotto la forma codificata del formato DICOM, peraltro in continua evoluzione, consentano di costruire quella che viene definita la "visualization pipeline", che in sostanza permette all'operatore finale di effettuare tutte quelle operazioni sopra indicate, senza che venga persa o alterata alcuna informazione. La particolare complessita' di un lavoro di questo genere deriva dalla miriade di strumenti informatici messi a disposizione sia dalle aziende commerciali produttrici e distributrici di software, che dai software messi

2 Il Medical Imaging

Nei due ultimi decenni sono stati sviluppati diversi tipi di sistemi medicali non invasivi, atti all'acquisizione di immagini tridimensionali degli organi interni di persone viventi. L' interpretazione di tali immagini, non e' tuttavia banale ne' deterministica, pertanto intorno a questa materia si e' organizzata una tematica scientifica che in terminologia anglosassone viene definita "medical imaging" ed in italiano con una piu' precisa connotazione "diagnostica per immagini".

La materia e' molto complessa, in quanto si tratta di far convergere in un filone di conoscenze sinergiche, gli apporti provenienti dalle conoscenze mediche, fisiologiche, fisiche, chimiche e cosi' via, in continua evoluzione per consentire la corretta diagnosi dei fenomeni osservati o, per lo meno, quella piu' aderente alla realta'.

2.1 Il formato DICOM

Con l'introduzione della tomografia assiale computerizzata e la successiva introduzione della risonanza magnetica si e' sentita l'esigenza di definire dei metodi standard per il trasferimento delle immagini. L'American College Radiology e la National Electrical

Manufactureres hanno formato un comitato per lo sviluppo dello standard *Digital Imaging and COmmunications in Medicine* (DICOM) che, sviluppato da diverse organizzazioni di standardizzazione, quali CEN TC251, IEEE, HL7 e ANSI USA, si e' velocemente ed ampiamente diffuso fra i produttori di apparecchiature medicali. Tale standard regolamenta tutte le problematiche connesse con il trattamento di immagini a partire dalla loro rappresentazione fino ad arrivare ai protocolli di comunicazione necessari per lo scambio (fra apparecchiature medicali o fra strutture ospedaliere) delle stesse.

Il formato DICOM e' costituito da un header contenente campi liberi, definiti dal *DICOM dictionary*, e da un corpo utilizzato per l'archiviazione di una o piu' immagini.

Il *DICOM dictionary* e' definito da ciascuna casa produttrice di apparecchiature medicali ed ognuna di esse utilizza l'header DICOM secondo le proprie esigenze.

Gli standard utilizzati per la compressione di tali immagini includono JPEG, LZW(Lempel Ziv Welch) e RLE(Run-length encoding). Lo standard DICOM e' in continua evoluzione ed e' aggiornato sotto le indicazioni del *Proceduresof the DICOM Standards Committee*.

2.2 Analisi dei dati

L'analisi delle immagini medicali prevede l'applicazione di due tecniche fondamentali: *segmentazione e registrazione*.

La segmentazione e' il processo che consente di identificare e quindi classificare i dati di un'immagine digitale.

Le tecniche di segmentazione possono essere suddivise in due categorie:

- Tecniche di estrazione di regioni (ossa, parti molli, ecc.): sfogliatura (threshold) e *region growing*.
- Tecniche di estrazione di contorni: *marching cubes*, modelli deformabili *snakes*, modelli deformabili *baloon*.

La sogliatura e' una operazione di selezione in base ai toni di grigio. E' estremamente rudimentale e come tale di semplice implementazione, inoltre non prevede la possibilita' di considerare un qualche criterio di connettivita'.

Il *region growing* e' una tecnica che prende in considerazione le caratteristiche dell'immagine per raggruppare i voxel e formare regioni secondo un criterio di omogeneita'. Partendo da alcune regioni iniziali (fase di suddivisione), queste vengono fuse (fase di fusione) per costituire regioni piu' estese, fino a quando il procedimento sia possibile.

Il *marching cubes* e' una tecnica per la costruzione di isosuperfici 3D. L'algoritmo originale e' stato proposto da Lorensen e Cline e prevede due passi:

1. si individua la superficie corrispondente a un determinato valore e si creano i triangoli;
2. per assicurare la qualita' dell'immagine si calcolano le normali alla superficie nei vertici di ogni triangolo, in modo da poter applicare ad esempio un *gouraud shading*.

La strategia complessiva e' di tipo *divide et impera*. Dapprima si localizza la superficie in un cubo logico di otto voxel, presi a gruppi di quattro da due slices adiacenti, poi si determina come la superficie interseca il cubo e poi si passa al successivo. Dato che ci sono otto vertici per ogni cubo e due stati (interno ed esterno) esistono 256 possibilita' che vengono codificate in un'apposita tavola. In realta' i casi base sono 14 e i restanti costituiscono permutazioni ottenibili per simmetria e rotazione. Si arriva a una triangolazione all'interno di ciascun cubo e si applica un'interpolazione lineare per trovare le intersezioni della superficie sugli spigoli.

Le *snakes* sono state introdotte da Kass per modellare e segmentare oggetti in immagini 2D.

Il *balloon* consiste in una geometria da deformare fino a coincidere con la superficie di un oggetto, ad esempio l'approssimazione poligonale di una sfera.

La registrazione e' il processo che consente di determinare una corrispondenza tra dati non omogenei. Qualora le immagini presentino problemi di allineamento, dovuti a movimenti del paziente o a tecniche di digitalizzazione manuali, si rende necessario un passo preliminare di *image registration* per il quale sono oggi disponibili programmi automatici o semiautomatici. Questo procedimento si avvale prevalentemente di algoritmi di tipo *optical flow* (features tracking) o *mutua informazione*. La registrazione, come sopra descritta, di immagini quale mezzo per risolvere i problemi di allineamento delle slices, ha un campo di applicazione piu' vasto e infatti le procedure per la ricostruzione dei tessuti molli, che e' di primaria importanza nella diagnostica medica, possono essere viste come particolari casi di registrazione.

La registrazione puo' rappresentare un processo costoso in termini di complessita' computazionale. Alcuni sistemi mettono a disposizione tecniche di *multiresolution registration* ampiamente utilizzate per migliore l'accuratezza e per ottimizzare la resa del risultato finale del processo di registrazione.

Con tale tecnica, gli algoritmi di registrazione sono eseguiti dapprima nei punti in cui l'immagine presenta un minore numero di pixel significativi, con un livello di accuratezza molto basso e i passi di elaborazione sono iterati fino a raggiungere il massimo livello di dettaglio la' dove l'immagine presenta un numero maggiore di pixel significativi.

L'applicazione realizzata, come successivamente verra' descritto, rende disponibili alcune funzionalita' relative al processo di segmentazione, mentre, al momento ancora non e'stata ultimata l'implementazione di funzionalita' relative alla registrazione.

Dal un punto di vista strettamente grafico, la segmentazione, con l'applicazione di algoritmi avanzati, consente di realizzare modelli tridimensionali di alto contenuto realistico, mentre la registrazione fornisce all'utente finale strumenti di diagnosi avanzata.

Le applicazioni attualmente in commercio per la visualizzazione e l'elaborazione delle immagini medicali, costituiscono delle scatole nere, il cui contenuto e'noto solo alle case produttrici di apparecchiature specifiche per la diagnosi medica.

Esistono inoltre una serie di prodotti open source, che offrono le medesime funzionalita', vincolate pero' all'utilizzo di un sistema operativo specifico, come ad esempio OsiriX, che con le sue librerie, quali *QuickTime*6 e *Cocoa*7, risulta essere fortemente legato alla piattaforma MacOS X.

Scopo dell'applicazione realizzata e' quindi quello di fornire uno strumento open source e cross platform per la visualizzazione e l'elaborazione di immagini medicali ed, inoltre, performante e scalabile, in modo da consentire la realizzazione, negli sviluppi futuri, di applicazioni per la visualizzazione medicale altamente immersiva.

Un'architettura astratta che soddisfi i requisiti dell'applicazione e' mostrata in figura 1. I componenti da utilizzare sono le librerie di alto livello, suddivise in librerie per la gestione dell'interfaccia grafica, librerie per la visualizzazione di immagini e modelli tridimensionali ed, infine, librerie per l'image processing.

Quest'ultimo componente, in particolare, deve prevedere il supporto per algoritmi di segmentazione e registrazione ed il supporto per l'I/O di documenti DICOM.

Alcune delle librerie open source e cross platform, oggi disponibili per la realizzazione dell'interfaccia grafica, comprendono *wxWindows*12, *FLTK*13, *Fox-Toolkit* 14, ecc.

Ognuna di queste e' corredata da una buona documentazione ed e' tenuta in continuo sviluppo dai rispettivi gruppi. La scelta e' quindi caduta su FLTK poiche' risulta essere molto semplice da utilizzare e allo stesso tempo completo.

Per quanto riguarda, invece, le librerie di visualizzazione e di image processing, la scelta e' ricaduta immediatamente su VTK e ITK. Questi due componenti, infatti, sono ampiamente utilizzati nel settore del medical imaging e risultano essere uno standard de facto per tali applicazioni. Il software da realizzare deve, quindi, integrare le funzionalita' offerte da tali librerie, costituendo un ambiente aperto per l'integrazione degli altri toolkit sopra citati ai fini di offrire un valido strumento per le applicazioni di realta' virtuale altamente immersive.

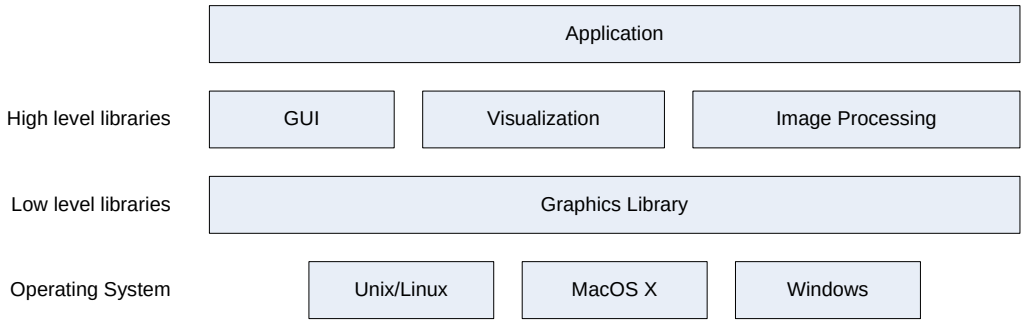

Figura 1 Architettura per applicazioni di Medical Imaging

3 Analisi e progettazione dell'applicazione

Nello scenario del Medical Imaging tutte le informazioni generate dai diversi sistemi di scansione di oggetti tridimensionali, sono raccolte in documenti di formato DICOM. Le informazioni di partenza, quindi, possono essere costituite da una o piµu immagini. Il sistema da realizzare deve poter fornire un'interfaccia utente che consenta all'operatore di poter visualizzare ed elaborare le immagini di partenza, costruire un modello tridimensionale ed esportarlo nel formato OBJ, nonche' di utilizzare gli strumenti di sviluppo da realizzare in questo lavoro per applicazioni future.

L'interfaccia utente deve essere necessariamente di facile comprensione anche per categorie di persone con un grado di preparazione informatica molto bassa. Allo stesso tempo, le funzionalita' di visualizzazione ed elaborazione, devono essere strutturate in maniera tale da consentire una continua evoluzione del software da realizzare.

Si utilizzera' il linguaggio UML per rappresentare i diagrammi che costituiscono i casi d'uso, i diagrammi delle classi individuate e i diagrammi di sequenza.

3.1 Requisiti funzionali

Qui vengono formalizzati i requisiti funzionali dell'applicazione. I diagrammi dei casi d'uso sono il punto di d'inizio della fase di analisi durante la progettazione di un sistema e vengono utilizzati per enumerare i requisiti di un sistema in modo che chiunque sia coinvolto con il sistema li possa comprendere.

L'identificazione dei requisiti del sistema viene svolta nella prima fase del Processo Unificato.

3.1.1 Diagramma dei casi d'uso

L'utente finale deve poter visualizzare ed elaborare le immagini DICOM attraverso un'interfaccia utente di facile comprensione.

Per visualizzare un file DICOM, l'utente deve chiaramente avere a disposizione un'interfaccia che gli consenta di selezionare un documento DICOM e poiche' il documento puo' essere costituito da una o piu' slice, deve avere a disposizione un'interfaccia che gli consenta di selezionare l'immagine da visualizzare. L'utente inoltre deve poter accedere all'header del documento DICOM per la lettura delle informazioni che costituiscono il *DICOM dictionary*. Aperto il documento, esso deve essere aggiunto ad una libreria dei documenti contenente la slice o le slices dei file DICOM aperti. L'utente quindi deve avere a disposizione un controllo che gli consenta di selezionare i diversi documenti aperti durante l'esecuzione dell'applicazione e deve poter chiudere ciascun elemento della libreria. Le operazioni che possono essere effettuate in fase di visualizzazione, includono la rotazione, lo zoom, la traslazione e l'impostazione dei livelli di grigio per esaltare particolari dell'immagine. Il visualizzatore da realizzare deve, inoltre, consentire l'interazione con modelli tridimensionali. Deve quindi essere messa a disposizione un'interfaccia che consenta all'utente l'interazione in uno spazio 3D. Nel diagramma in figura 2 sono rappresentati i casi d'uso che determinano i requisiti per le funzionalita' di visualizzazione.

I casi d'uso "Imposta livelli di grigio" e "Mostra header" sono stati realizzati con una relazione di estensione, poiche' rappresentano funzionalita' di uso facoltativo e quindi non necessarie per espletare le funzionalita' previste dal caso d'uso "Visualizza immagini DICOM e modelli 3D". Di seguito sono descritte con maggiore dettaglio, attraverso le specifiche testuali, le funzionalita' richieste e sono definite le azioni che il sistema deve eseguire a seguito dell'attivazione di ciascun caso d'uso.

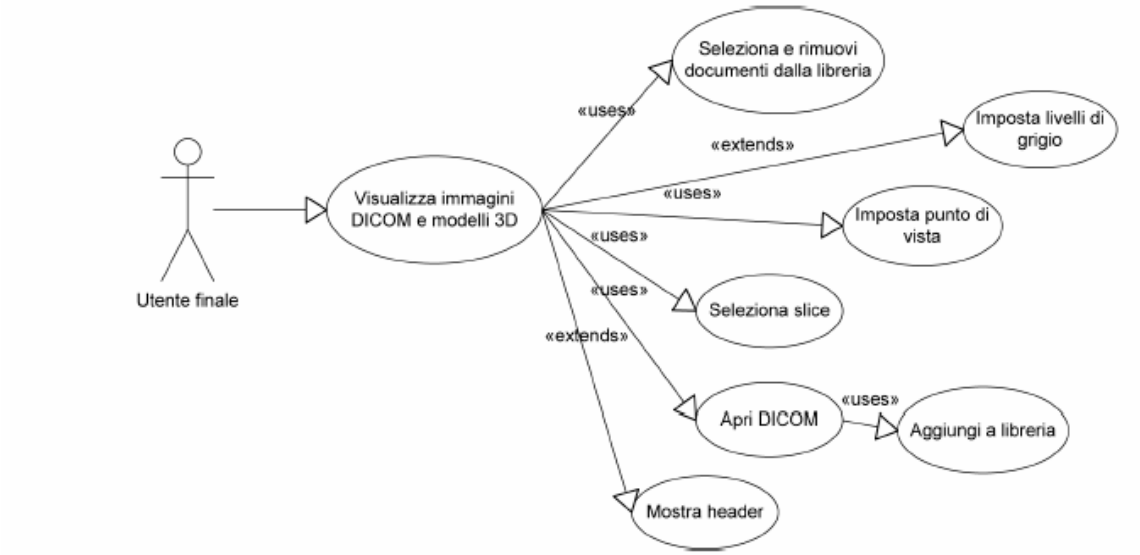

Figura 2 Casi d'uso: utente finale, visualizzazione

Visualizza immagini DICOM e modelli 3D

Breve descrizione Questo caso d'uso permette all'utente finale di visualizzare un immagine DICOM e un modello 3D.

Attori utente finale.

Precondizioni L'utente utilizza un'interfaccia grafica che gli consente di selezionare un file DICOM o una directory contenente una serie di file DICOM.

Flusso principale Il caso d'uso inizia quando l'utente chiede al sistema di mostrare i file e le directory del filesystem. L'utente seleziona il documento da visualizzare. Il sistema visualizza la prima slice della directory selezionata o il singolo file selezionato.

Flussi alternativi Il caso d'uso e' attivato dal sistema a seguito della costruzione di un modello tridimensionale. In questo caso il sistema mette a disposizione i controlli per un visualizzatore di modelli 3D.

Postcondizioni Se il caso d'uso ha avuto successo il documento viene aggiunto alla libreria dei documenti aperti.

Imposta livelli di grigio

Breve descrizione Consente all'utente di modificare la scala dei livelli di grigio.

Attori utente finale.

Precondizioni Il caso d'uso “Visualizza immagini DICOM e modelli 3D" deve essere stato gia' attivato almeno una volta.

Flusso principale Il caso d'uso viene attivato quando l'utente utilizza degli appositi controlli per la regolazione dei livelli. Il sistema modifica i livelli di grigio in funzione dei valori selezionati dall'utente.

Imposta punto di vista

Breve descrizione Questo caso d'uso consente all'utente di modificare i parametri di zoom, rotazione e panning dell'immagine o del modello tridimensionale. In quest'ultimo caso deve consentire anche di spostare il punto di vista del modello in uno spazio 3D.

Attori utente finale.

Precondizioni Il caso d'uso “Visualizza immagini DICOM e modello 3D" deve essere stato gia' attivato almeno una volta.

Flusso principale Il caso d'uso viene attivato quando l'utente mediante tastiera e mouse modifica i parametri di visualizzazione. Il sistema, in ascolto per tali eventi, elabora le immagini e/o il modello 3D e visualizza i risultati.

Seleziona slice

Breve descrizione Questo caso d'uso consente all'utente di selezionare la slice da visualizzare.

Attori utente finale.

Precondizioni Il caso d'uso “Visualizza immagini DICOM e modello 3D" deve essere stato gia' attivato almeno una volta e l'utente deve aver scelto una directory contenente una serie di file DICOM.

Flusso principale L'utente mediante controlli di tipo slide bar e bottoni, seleziona la slice da visualizzare. Il sistema visualizza il risultato della scelta.

Apri DICOM

Breve descrizione Il sistema offre un'interfaccia per la selezione di un file DICOM o di una directory contenente file DICOM.

Attori utente finale.

Flusso principale L'utente genera un evento per l'esecuzione del caso d'uso. Il sistema mostra l'interfaccia di selezione.

Aggiungi a libreria

Breve descrizione Il sistema aggiunge alla libreria del documenti DICOM aperti ciascun file selezionato dall'utente.

Attori utente finale.

Precondizioni Il caso d'uso “Apri DICOM" deve essere stato attivato.

Flusso principale Il sistema aggiunge ad lista dei documenti aperti l'ultimo elemento selezionato dall'utente. A tale elemento viene associato un nome che consente all'utente di poter successivamente identificare il file aperto. Se l'utente seleziona un serie DICOM, il sistema aggiunge al nome dell'elemento, il numero di slices appartenenti a quest'ultimo.

Seleziona e rimuovi documenti dalla libreria

Breve descrizione Il sistema mette a disposizione un'interfaccia che consente all'utente di poter rimuovere o selezionare i documenti DICOM aperti nel corso dell'esecuzione dell'applicazione.

Attori utente finale.

Precondizioni Il caso d'uso “Aggiungi a libreria" deve essere stato attivato.

Flusso principale L'utente mediante controlli di tipo menu a tendina e bottoni, seleziona il documento da visualizzare. Il sistema visualizza il risultato della scelta e ripristina lo stato di visualizzazione del documento (Livelli di grigio, slice corrente, ecc.).

Mostra header

Breve descrizione L'utente puo’ visualizzare i campi dell'header del file DICOM in un documento di tipo testuale.

Attori utente finale.

Precondizioni Il caso d'uso “Visualizza immagini DICOM e modello 3D" deve essere stato gia’ attivato almeno una volta.

Flusso principale L'utente genera un evento per l'esecuzione del caso d'uso. Il sistema mostra le informazioni testuali contenute nel documento DICOM.

Le funzionalita’ di elaborazione delle immagini, rappresentate in figura 3, costituiscono solo le funzionalita’ da implementare per questo elaborato e devono risultare da esempio per l'applicazione di filtri, implementati nelle librerie VTK ed ITK, per successivi sviluppi dell'ambiente.

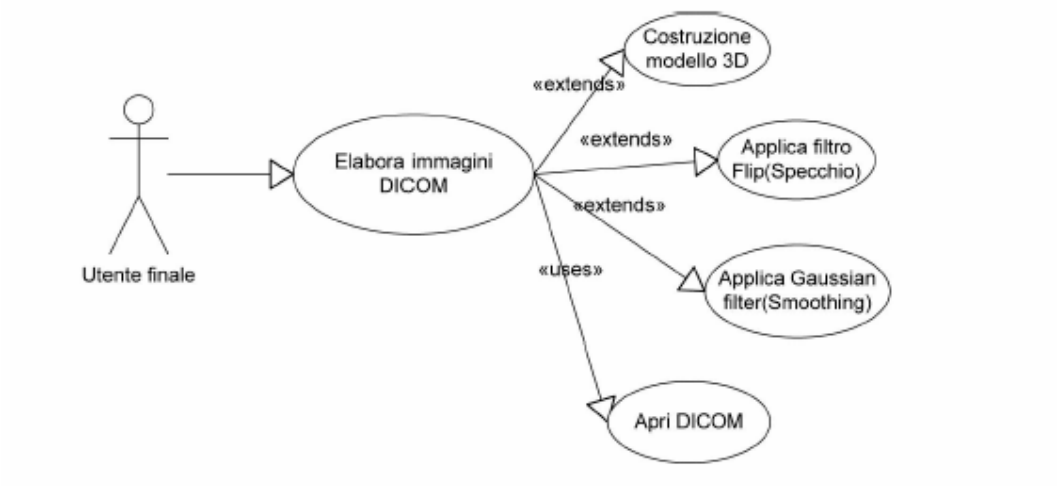

Figura 3 Casi d'uso: utente finale, elaborazione delle immagini

L'utente finale deve poter applicare filtri di flipping, ovvero di poter “ribaltare" un'immagine lungo un determinato asse, e filtri di tipo gaussiano per lo smoothing delle immagini. Quest'ultimo filtro risulta essere un comodo strumento per migliorare il risultato finale della ricostruzione 3D.

Tali funzionalita’ rappresentano degli strumenti facoltativi per l'elaborazione delle immagini e pertanto sono in relazione di estensione con il caso d'uso “Elabora immagini DICOM".

La ricostruzione tridimensionale del modello 3D necessiterebbe l'implementazione di algoritmi di segmentazione avanzati, per definire il significato di regioni dell'immagine che costituiscono parti molli, rigide, come vasi sanguigni, ossa, ecc. per una corretta ricostruzione del modello tridimensionale.

Nella applicazione realizzata, la tecnica di ricostruzione utilizzata e’ basata su algoritmi di *contouring* e, pertanto, l'applicazione di filtri di smoothing dell'immagine, risulta estremamente funzionale. Anche per queste funzionalita’ risulta chiaramente indispensabile offrire un'interfaccia per la selezione del documento DICOM da elaborare. I filtri di elaborazione dell'immagine, invece, devono agire sull'intero documento, questo implica che la selezione della slice e’ una funzionalita’ non determinante in questo caso.

Di seguito la descrizione dettagliata dei principali casi d'uso.

Costruzione modello 3D

Breve descrizione Questo caso d'uso consente all'utente di costruire un modello 3D applicando un algoritmo di tipo contour con le tecniche dei *marching cubes*.

Attori utente finale.

Precondizioni Il caso d'uso “Visualizza immagini DICOM e modello 3D" deve essere stato gia’ attivato almeno una volta e l'utente deve aver scelto una directory contenente una serie di file DICOM.

Flusso principale Il caso d'uso inizia quando l'utente, impostati i parametri dell'algoritmo di contour, chiede al sistema di realizzare un modello tridimensionale. Il sistema genera il modello e mette a disposizione i controlli per la visualizzazione di modelli tridimensionali.

Applicazione di filtri gaussiani e flipping

Breve descrizione	Questi casi d'uso consentono all'utente di effettuare operazioni di flipping e di applicare un filtro di tipo gaussiano all'immagine.
Attori	utente finale.
Precondizioni	Il caso d'uso “Visualizza immagini DICOM e modello 3D" deve essere stato gia’ attivato almeno una volta.
Flusso principale	Il caso d'uso inizia quando l'utente chiede al sistema di applicare tali filtri al documento visualizzato. Il sistema chiede all'utente di impostare i parametri di ciascun filtro e successivamente mostrera’ i risultati ottenuti.

3.1.2 Classi individuate

In questa paragrafo sono trattati gli elementi fondamentali utilizzati per rappresentare staticamente oggetti e funzionalita’ dell'intero sistema, ovvero, le classi.

L'analisi dei requisiti ha portato alla realizzazione del diagramma delle classi mostrato in figura 4. Tale diagramma rappresenta l'architettura software utilizzata per il sistema da realizzare. Le relazioni, descritte da una linea che collega due classi, sono di tipo strutturale. Questo implica che se una classe e’ in relazione con un'altra, quest'ultima costituira’ un attributo della prima. Le relazioni di specializzazione, invece, sono descritte da una linea con una freccia. La classe verso cui punta la freccia e la superclasse, o classe padre, ovvero la classe che viene specializzata. Nel diagramma e’ presente inoltre una relazione di aggregazione tra le classi imageViewer e dicomImage. Tale relazione indica che la classe imageViewer e’ costituita da piµu parti, le classi che formano le parti, pero’, al contrario della relazione di “composizione", possono esistere, ovvero possono essere istanziate come oggetti, anche senza la classe complessiva.

I metodi e gli attributi saranno discussi successivamente per le principali classi progettate.

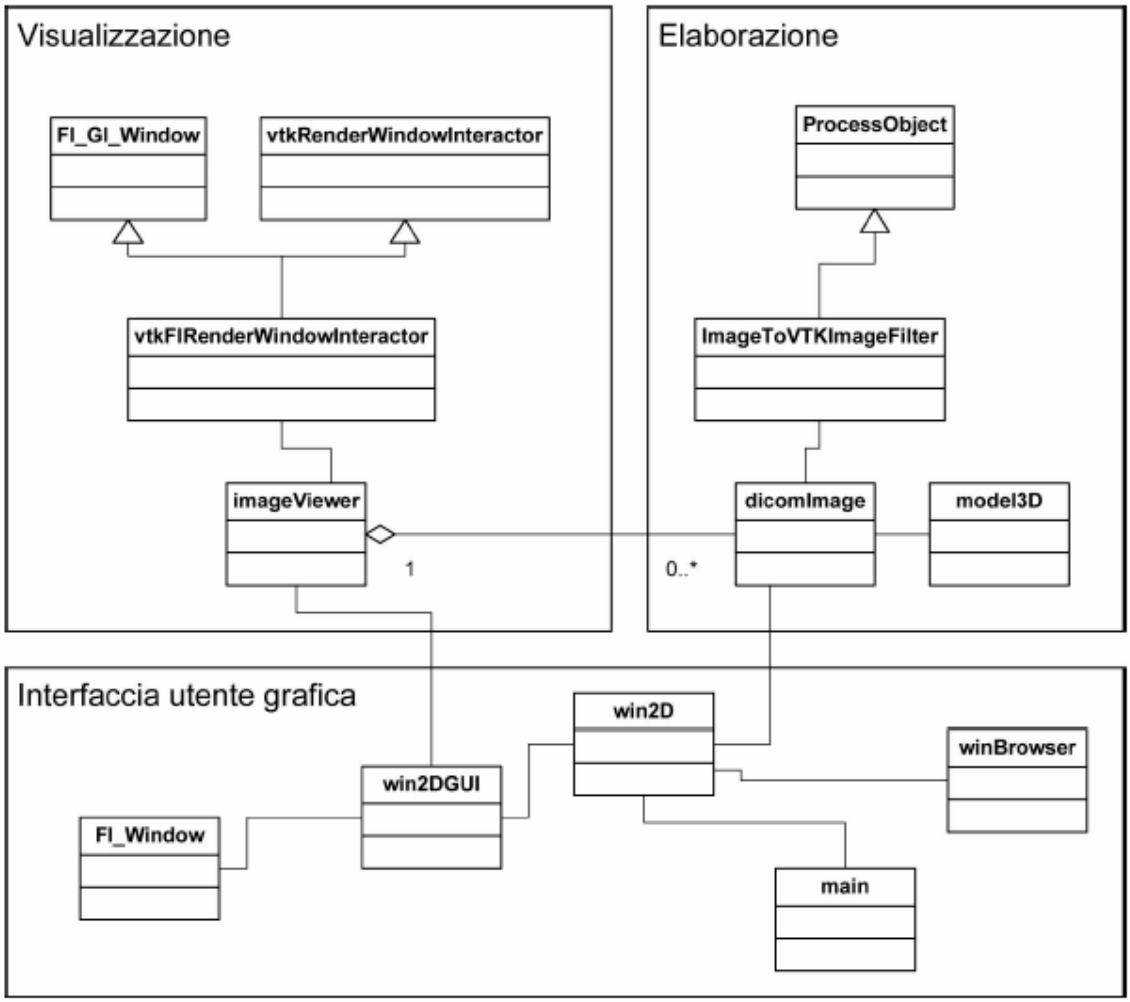

Figura 4 Diagramma delle classi

E' opportuno sottolineare che le classi ProcessObject, Fl Gl Window, Fl Window e vtkRenderWindowInteractor, sono rispettivamente implementate nei toolkit ITK, FLTK e VTK. Inoltre altre classi dei rispettivi toolkit utilizzate in questo progetto non sono rappresentate nel diagramma. La classe main e' rappresentata come classe ma in effetti costituisce semplicemente il punto di partenza dell'esecuzione dell'applicazione e in C++ non risulta essere una classe bensì una funzione.

Nel diagramma delle classi e' possibile notare che il sistema e' stato suddiviso in tre blocchi principali: visualizzazione, elaborazione e interfaccia grafica utente.

Il primo sottosistema e' chiaramente responsabile delle operazioni di visualizzazione dei documenti DICOM. Tali documenti sono rappresentati mediante la classe dicomImage del sottosistema di elaborazione.

Poiche' nei requisiti funzionali e' stata messa in evidenza l'esigenza di poter aprire e lavorare su piu' documenti DICOM, ciascuno contenente una o piu' slice, il sottosistema di visualizzazione e' in relazione di aggregazione con il sottosistema di elaborazione.

Il sottosistema GUI e' in relazione sia con il visualizzatore che con il sottosistema di elaborazione.

La classe win2DGUI, infatti, responsabile della creazione della GUI, oltre alla creazione dei bottoni, dei menu, e degli altri widget, inizializza il visualizzatore imageViewer, pertanto quest'ultimo e' rappresentato come attributo della classe win2DGUI.

La classe win2D ha come responsabilita’ quella di offrire le funzionalita’ scaturite dall'attivazione di eventi della GUI. Pertanto, le istanze della classe dicomImage, sono create proprio da questa classe e vengono aggiunte alla libreria dei documenti DICOM aperti, gestita dall'oggetto imageViewer.

Diamo ora una descrizione delle classi con particolare riferimento alle relazioni e alle responsabilita’ di ciascuna.

3.2.1 Gestione delle immagini DICOM

Le funzionalita’ di lettura e di elaborazione delle immagini DICOM sono affidate alla classe dicomImage.

Il toolkit ITK, che consente la lettura di documenti DICOM, non mette a disposizione alcuna classe per la visualizzazione. Per effettuare questa operazione, infatti, e’ necessario utilizzare il toolkit VTK. Poiche’ inoltre risulta necessario per sviluppi futuri utilizzare algoritmi di segmentazione e registrazione, risulta indispensabile realizzare una conversione dei dati ITK nel formato VTK.

Tale conversione deve avvenire dinamicamente cosicche’ una volta acquisita un'immagine nel formato VTK, sia ancora possibile applicare modifiche alla pipeline di ITK. E' necessario quindi realizzare un filtro che consenta di collegare le due pipeline. La classe itkImageToVTKImageFilter, di figura 5, costituisce il filtro che consente di collegare dinamicamente le due pipeline.

Figura 5 Class diagram: itkImageToVTKImageFilter

Essa specializza la classe ProcessObjects che, come visto nel capitolo secondo, costituisce la classe operante su dati e produce nuovi data objects. Il filtro utilizza le classi VTKImageExporter, appartenente all'insieme delle classi di ITK, e vtkImageImport, classe di VTK, che consentono rispettivamente di esportare le strutture dati di ITK nel formato VTK, rappresentate come array, e di importare tali strutture come classe vtkImageData.

Le responsabilita’ della classe itkImageToVTKImageFilter sono quindi limitate alla gestione dell'esecuzione della pipeline che deve consentire un livello di astrazione tale da rendere trasparente questa conversione. La pipeline risultante dall'integrazione della pipeline di ITK e VTK e’ mostrata in figura 6.

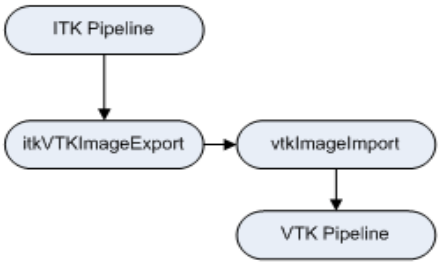

Figura 6. Fusione pipeline ITKVTK

Il filtro itkImageToVTKImageFilter e' utilizzato dalla classe dicomImage, quindi identificato dall'attributo m_itkConnector. I metodi loadDicom sono responsabili del caricamento delle immagini DICOM e della conversione di tali dati nel formato VTK.

L'utente deve poter aprire un'immagine DICOM o una directory contente piu' immagini. I metodi loadDicom, pertanto, sono stati realizzati per le diverse esigenze e sfruttano le funzionalita' dell'overloading della programmazione ad oggetti. Le immagini sono allocate nelle strutture dati itk::Image e vtk::vtkImageData, identificate rispettivamente dagli attributi m_itkImPointer e m_vtkImageData, mentre l'header viene memorizzato in una struttura dati a cui e' possibile accedere mediante l'attributo m_dicomHeader, che puo' essere rappresentato come una matrice di stringhe contenente i campi *chiave* e *valore.*

I metodi getHeaderLength, getHeaderKey e getHeaderValue rappresentano l'interfaccia per l'accesso pubblico alle informazioni contenute dell'header DICOM. I metodi transformImage e gaussianFilter rappresentano un'interfaccia alle classi di VTK per il fl*ipping* delle immagini e per l'utilizzo di filtri gaussiani.

I metodi getModel3d e renderModel3d rappresentano invece un'interfaccia alla classe model3d di cui si discutera' nel prossimo paragrafo.

Altri attributi della classe dicomImage specificano lo stato di visualizzazione dell'immagine. Ad esempio il metodo setColorWindow consente di assegnare all'attributo m_colorWindow un valore di tipo *float* che specifica l'impostazione dei livelli di grigio. L'implementazione degli algoritmi necessari per ottenere una variazione di luminanza o ancora rotazione, scala e panning delle immagini e' a carico della classe imageViewer di cui si discutera' successivamente.

3.2.2 Ricostruzione tridimensionale

La classe model3D, di figura 7, e' responsabile della ricostruzione del modello tridimensionale che puo' essere realizzato a partire da una serie di immagini DICOM. Il costruttore prende come parametro di ingresso un'immagine vtkImageData, identificata dall'attributo p_imageData, ed il metodo filterContour e' responsabile della ricostruzione tridimensionale del modello, identificato dall'attributo M_actor, con l'ausilio di algoritmi di estrazione dei contorni implementati nel toolkit VTK. Gli altri attributi consentono di identificare gli oggetti creati per la realizzazione della pipeline, mostrata e chiarita nel capitolo successivo, per la ricostruzione 3D del modello. Ulteriori filtri di ricostruzione dovranno essere implementati in questa classe negli sviluppi futuri dell'applicazione. Il metodo objExporter e' invece responsabile del salvataggio dei modelli tridimensionali, rappresentati dalla struttura vtkActor, nel formato OBJ della Alias Wavefront.

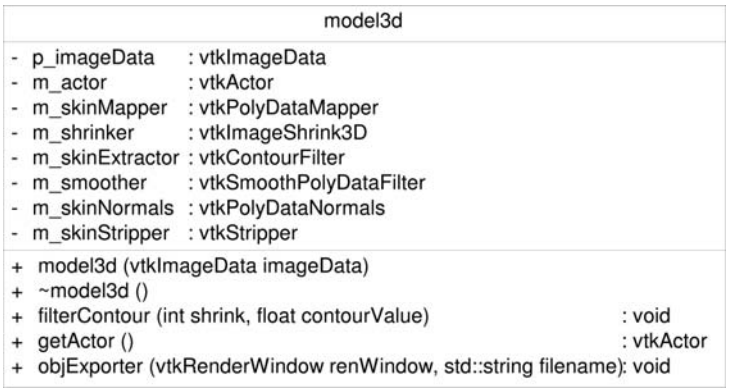

Figura 7 Class Diagram:model3d

3.2.3 Visualizzazione 2D e 3D

Il visualizzatore da realizzare per l'ambiente di sviluppo deve consentire di offrire un'interfaccia flessibile e completa per l'interazione con immagini e modelli tridimensionali. La classe imageViewer, il cui diagramma UML e' mostrato in figura 8, e' responsabile della visualizzazione e dell'interazione con uno o piu' oggetti della classe dicomImage.

Tale classe inoltre deve consentire l'integrazione del visualizzatore VTK in una GUI realizzata con il toolkit FLTK.

Figura 8 Class Diagram:imageViewer

Tale funzionalita' e'offerta dalla classe vtkFlRenderWindowInteractor di figura 9. Quest'ultima risulta essere una specializzazione della classe Fl_Gl_Window offerta da FLTK e della classe, implementata in VTK vtkRenderWindowInteractor.

La classe imageViewer istanzia tutti gli oggetti che costituiscono la graphics pipeline di VTK ed in particolare utilizza un oggetto di tipo vtkFlRenderWindowInteractor per l'interazione dell'utente con l'ambiente 2D o 3D. La flessibilita' nel visualizzare immagini 2D o modelli 3D e'

offerta proprio da quest'ultimo. VTK mette a disposizione delle classi ad-hoc per la visualizzazione delle immagini bidimensionali.

Tali classi pero' risultano poco efficienti e rendono particolarmente impegnativo lo sviluppo di un visualizzatore unico per immagini 2D e modelli 3D.

La strategia da utilizzare, quindi, e' quella di realizzare una graphics pipeline per modelli 3D e visualizzare le immagini come *texture* applicate su modelli tridimensionali. Questo consente di utilizzare al meglio le GPU (Graphic Processor Unit) che dispongono di un acceleratore 3D. Le operazioni di *panning*, *rotation* e *zoom*, possono essere infatti eseguite in *hardware*, con chiamate OpenGL, e non richiedono elaborazione da parte della CPU. Chiaramente l'interazione da parte dell'utente con un'immagine 2D o con un modello 3D e' differente. L'oggetto m_winInteractor della classe vtkFlRenderWindowInteractor, responsabile dell'interazione utentevisualizzatore, cambia le modalita' di interazione dinamicamente, ovvero in funzione della tipologia di dati da visualizzare. Altre responsabilita' della classe includono la selezione della slice da visualizzare, implementate nel metodo setZSlice, nel caso in cui l'utente abbia selezionato una serie di documenti DICOM da visualizzare, la gestione della libreria di documenti DICOM aperti, funzionalita' offerte mediante i metodi addDicomImage, getDicomImage e deleteAllDicomImage e infine l'impostazione dei livelli di grigio, mediante i metodi setColorLevel e setColorWindow.

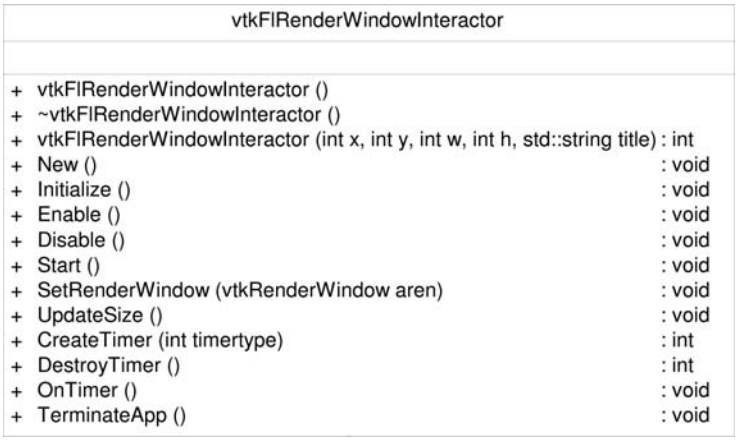

Figura 9 Class Diagram:vtkFlRenderWindowInteractor

3.2.4 Modellazione interfaccia grafica utente

Le classi responsabili della gestione dell'interfaccia grafica utente e, quindi, di parte dell'interazione tra utente finale e sistema sono: win2D, win2DGUI e winBrowser. In particolare la classe win2DGUI e' responsabile della creazione dell'interfaccia utente grafica e la gestione degli eventi e' affidata quindi alla classe win2D. Con FLTK, il toolkit utilizzato per la realizzazione della GUI, la gestione degli eventi e' basata su funzioni di callback. L'idea e' quindi di realizzare un metodo show, che si occupa del "disegno" della GUI, nella classe win2DGUI e di implementare i metodi per la gestione delle funzioni di callback nella classe win2D.

Cio' consente di limitare le responsabilita' di ciascuna classe e di realizzare un'astrazione tra implementazione dell'interfaccia utente e sistema.

Figura 10 Class Diagram:win2DGUI

Nel diagramma in figura 10 sono mostrati metodi e attributi della classe win2DGUI, in figura 11 quelli della classe win2D. Gli attributi della classe win2DGUI rappresentano i diversi controlli utilizzati per realizzare l'interfaccia: menu a tendina, bottoni e slide bar. La classe win2DGUI rappresenta quindi un attributo della classe win2D e gli elementi grafici, i *widget*, rappresentano gli attributi della classe win2DGUI.

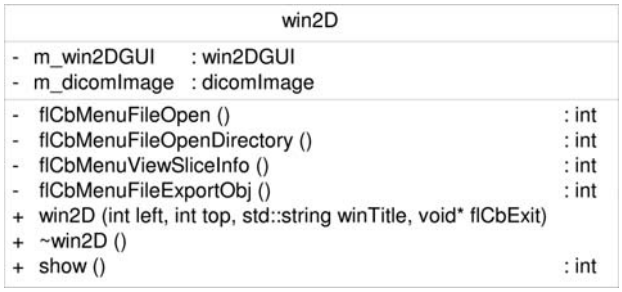

Figura 11 Class Diagram:win2d

Il visualizzatore di slice puo' essere considerato come un *widget* ed e' stato quindi definito come attributo della classe di creazione della GUI sotto il nome di m_imageViewer.

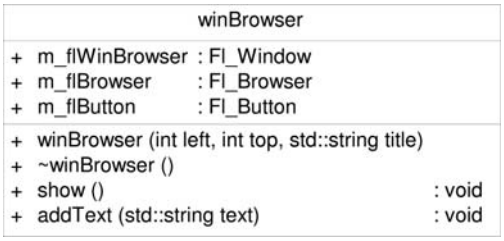

Figura 12 Class Diagram:winBrowser

La classe winBrowser, il cui diagramma e' mostrato in figura 12, ha il compito di visualizzare un browser testuale ed e' stata progettata per la visualizzazione delle informazioni contenute nell'header di un documento DICOM. Il *widget* Fl_Browser, rappresentato come attributo di tale classe, consente di mostrare informazioni di tipo testuale con l'ausilio di barre di scorrimento.

3.3 Scenari

Definite le principali funzionalita' e responsabilita' delle classi, in questo paragrafo si cerchera' di considerare le interazioni tra gli oggetti per eseguire i principali casi d'uso. Questo consente di comprendere a meglio come gli oggetti collaborino mediante l'invio di messaggi. Di seguito sono riportati alcuni scenari d'esempio rilevanti rappresentati con l'ausilio di diagrammi sequence.

Un diagramma di sequenza ha due componenti principali: gli oggetti attivi e le comunicazioni fra di essi. Gli *oggetti attivi* sono gli oggetti che svolgono una qualsiasi funzione all'interno del sistema, che siano istanze di classi o attori.

I *messaggi* scambiati fra i vari oggetti attivi sono la parte piu' importante dei diagrammi di sequenza e illustrano il flusso fra oggetti, il modo in cui gli oggetti interagiscono e quali condizioni possono modificare il flusso.

3.3.1 Visualizzazione immagini DICOM e modello 3D

In tale scenario l'utente esegue l'applicazione e chiede al sistema di visualizzare una singola immagine, vediamo, quindi con il supporto di un diagramma di sequenza, mostrato in figura 13, l'interazione tra utente e sistema e tra le classi istanziate. I passi possono essere tradotti in linguaggio naturale come segue:

1. L'utente finale esegue un'istanza dell'applicazione. Viene quindi invocato il main, il quale crea un'istanza della classe win2D, che a sua volta crea un oggetto win2DGUI. Quest'ultimo crea gli oggetti necessari per la realizzazione dell'interfaccia grafica ed in particolare crea un'istanza della classe imageViewer.
   I messaggi di figura includono: init, showGUI e showViewer.

2. L'utente finale chiede al sistema di fornire l'elenco dei file e delle directory del filesystem. A seguito di tale evento, catturato dall'oggetto win2D, il sistema mostra un'interfaccia per la selezione del file.
   Messaggi: fileOpen, showFileOpen, fileOpenWindow.

3. L'utente seleziona il file dcmFile.dcm.
   Messaggio: selectFile.

4. Il sistema crea un oggetto dicomImage e invoca i metodi necessari per l'apertura del file.

Messaggio: loadDicom.

5. Il sistema chiede all'istanza della classe imageViewer di aggiungere il nuovo oggetto dicomImage.
   Messaggi: getImageViewer e addDicomImage.

6. Il sistema infine chiede all'oggetto imageViewer di visualizzare l'immagine appartenente all'oggetto dicomImage creato al passo 4.
   Messaggi: showDicomImage e showImage

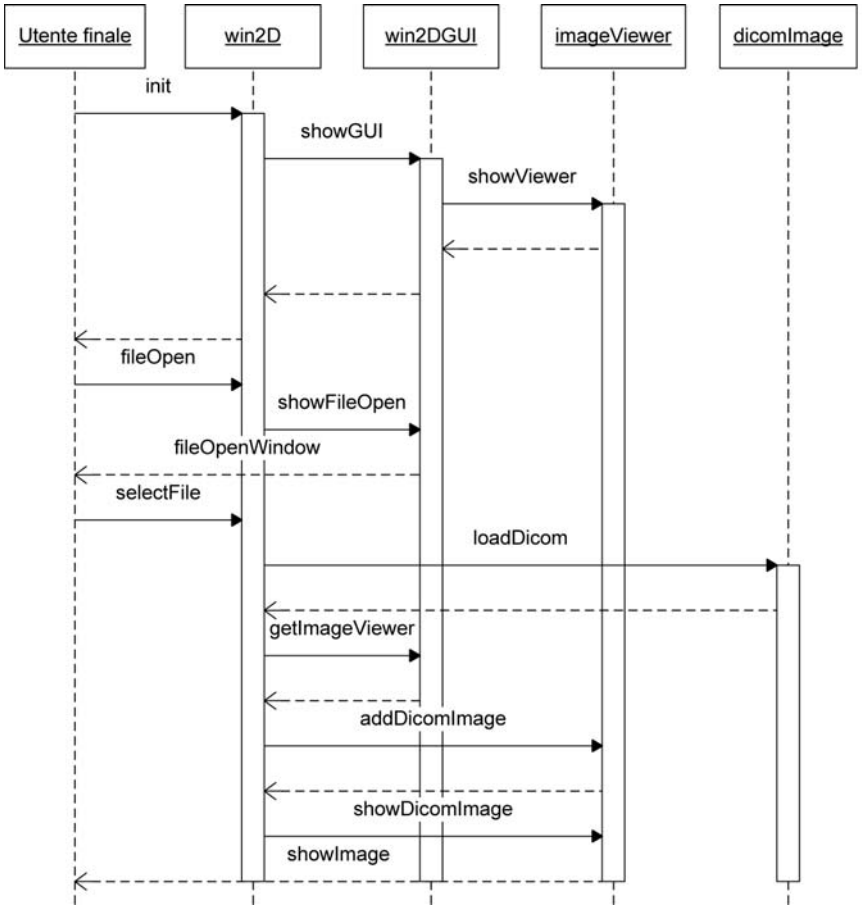

Figura 13 Sequence Diagram:visualizza immagine DICOM e modello 3D

3.3.2 Costruzione modello 3D

L'utente ha gia' richiesto al sistema di visualizzare una serie di slice. E chiede ora di costruire e visualizzare il modello tridimensionale ricostruito mediante un algoritmo contour. In figura 14 il relativo diagramma e, in linguaggio naturale, la sequenza dei passi di interazione e' la seguente:

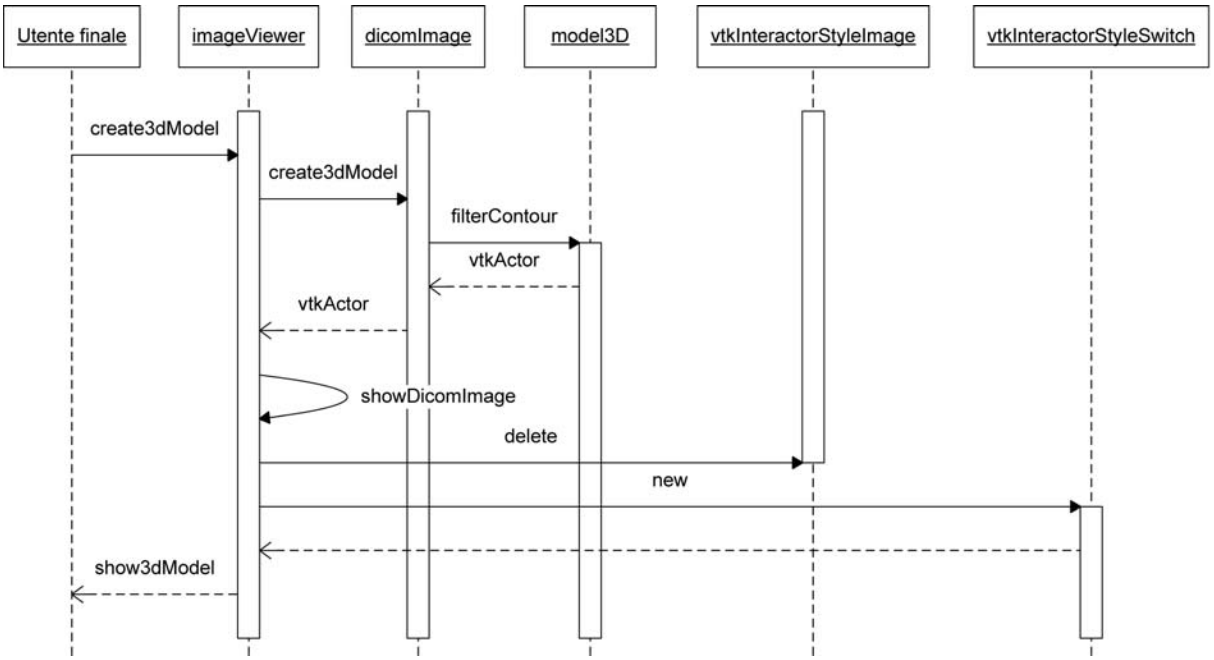

Figura 14 Sequence Diagram:Costruzione modello 3D

1. L'utente finale imposta i parametri necessari per l'esecuzione dell'algoritmo.
   Messaggio: create3dModel.

2. L'oggetto imageViewer chiede all'oggetto dicomImage di realizzare il modello 3D.
   Messaggio: create3dModel.

3.L'oggetto dicomImage crea un'istanza della classe model3D e invoca il metodo di ricostruzione del modello tridimensionale.
   Messaggi: filterContour, vtkActor.
4.L'oggetto imageViewer invoca il proprio metodo di visualizzazione, analizzando l'oggetto dicomImage, imposta i controlli di visualizzazione tridimensionale invocando il metodo delete dell'oggetto vtkInteractorStyleImage e creando un'istanza della classe vtkInteractorStyleSwitch che consente all'utente l'interazione in uno spazio tridimensionale.
   Messaggi: showDicomImage, delete, new, show3dModel.

4 Realizzazione dei componenti software

Tutti i componenti dell'applicazione sono stati implementati in C++, utilizzando come ambiente di sviluppo Microsoft Visual Studio ed il software *CMake*, open source e cross platform, per la generazione dei file di configurazione.
Trattandosi, inoltre, di un'applicazione cross platform sono state utilizzate solo funzionalita' dell'insieme del *C Standard Library* e classi *STL (Standard Template Library)*.

4.1 GUI e gestore degli eventi

I componenti software dell'applicazione realizzata, relativi alla GUI e al Gestore degli eventi, e le rispettive relazioni di inclusione sono mostrati nei diagrammi di figura 15. Nel main, il punto di partenza dell'applicazione, viene creata un'istanza della classe win2D ed e' invocato il metodo show dell'oggetto. Realizzato il contesto grafico, mediante un oggetto win2DGUI, nel metodo show dell'oggetto win2D e' invocata la funzione Fl::run() del toolkit FLTK. Tale funzione richiama a sua volta la funzione Fl::wait() ripetutamente finche' le finestre sono visualizzate.

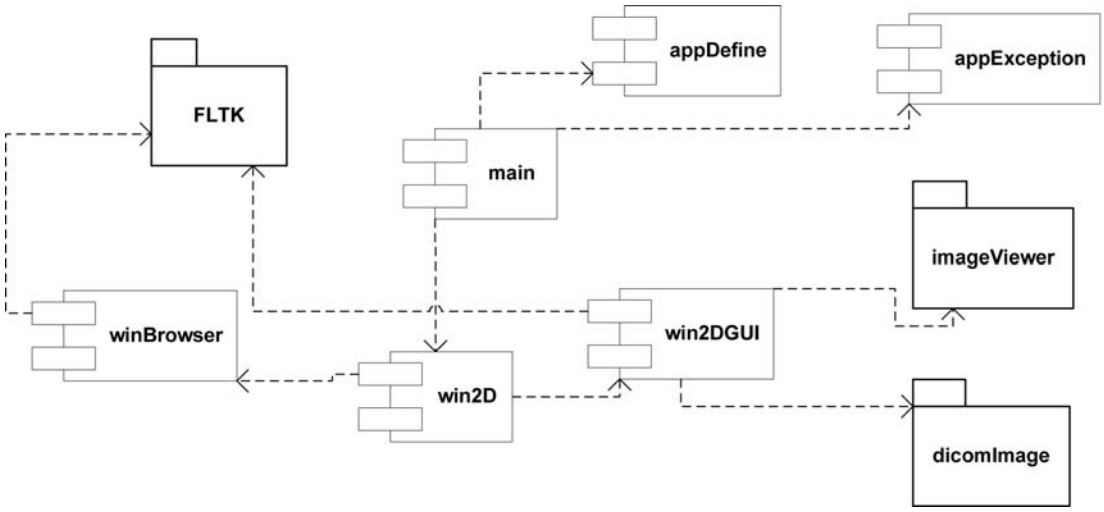

Figura 15 Component Diagram:GUI e gestore eventi

L'applicazione termina ritornando il valore della funzione Fl::run(). Le funzionidi callback sono state implementate nella classe win2D e nella classe imageViewer. Il gestore della GUI crea un'istanza della classe Fl_Window, che rappresenta la finestra nella quale disegnare i controlli, un'istanza della classe imageViewer, che rappresenta il visualizzatore 2D e 3D e infine crea ed i imposta i parametri per i diversi controlli.

4.2 Immagini DICOM e modello 3D

I componenti mostrati in figura 16 realizzano le funzionalita' di apertura, di elaborazione e di ricostruzione del modello tridimensionale. Per l'apertura dei file DICOM, cosi' come previsto in fase di progettazione, sono stati realizzati due metodi. Il primo responsabile dell'apertura di un singolo file, il secondo di una serie di slice. Le classi di ITK da utilizzare per tali operazioni, infatti, sono differenti, ma la struttura dati (vtkImageData) in cui sono contenute la singola immagine o la serie di slice, e' la stessa.

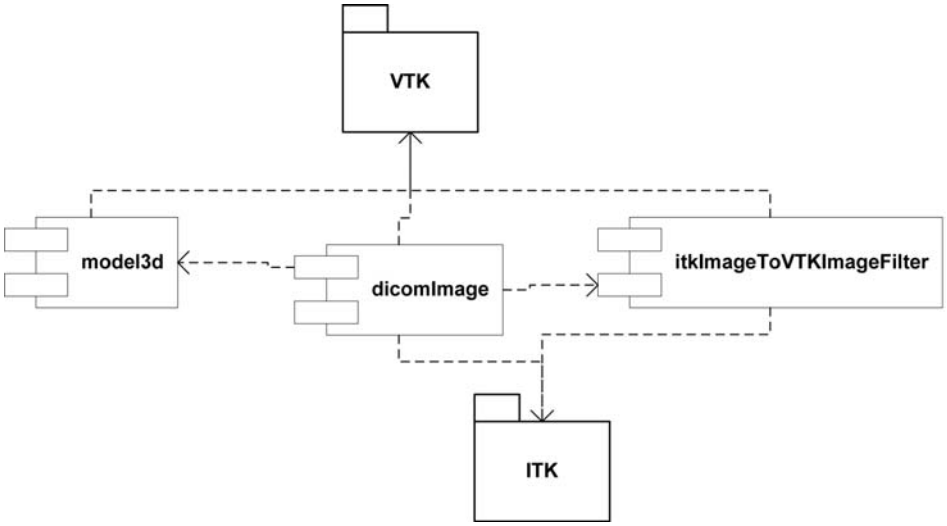

Figura 16 Component Diagram:dicomImage

La classe ImageIOType di ITK e' stata utilizzata per la lettura l'header del file DICOM ed e' stata realizzata una struttura dati, rappresentata come una matrice di stringhe, contenente tali informazioni. I filtri gaussiani e le operazione di flipping sono stati realizzati utilizzando classi di VTK, ma poiche' le pipeline di ITK e VTK sono congiuntamente fuse in un'unica pipeline mediante la classe itkImageToVTKImageFilter, sarebbe stato possibile utilizzare indistintamente uno dei due toolkit.
La ricostruzione del modello 3D e' stata realizzata, cosi' come espresso nei requisiti, utilizzando algoritmi di contour implementati in VTK. La classe dicomImage crea un oggetto model3D e invoca il metodo filterContour di quest'ultimo. Il metodo invocato realizza il modello tridimensionale e ritorna un oggetto vtkActor che rappresenta l'oggetto 3D e le proprieta' del materiale ad esso assegnato. La classe model3D, inoltre, utilizza la classe vtkOBJExporter per la conversione del modello 3D nel formato OBJ. In figura 17 e' riportata la pipeline per la ricostruzionedel modello 3D e, di seguito, il codice del metodo filterContour della classe model3D.

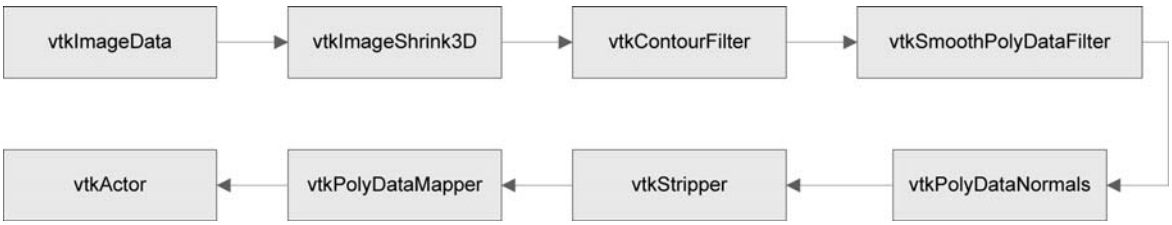

Figura 17 Pipeline per la costruzione del modello 3d

4.3 Visualizzazione 2D e 3D

La visualizzazione delle immagini DICOM e dei modelli tridimensionali e' gestita dai componenti mostrati in figura 18. La classe imageViewer come detto in fase di progettazione, funge da *widget* del toolkit FLTK. Questo componente, infatti, e' stato implementato per essere utilizzato all'interno di una finestra FLTK con l'ausilio della

classe vtkFlRenderWindowInteractor. Specializzando le classi Fl_Gl Window e vtkRenderWindowInteractor, infatti, l'oggetto invia i messaggi catturati da FLTK al toolkit VTK, in modo da rendere disponibile l'interazione con le classi di VTK utilizzate per la visualizzazione, poiche' l'handle del contesto GL richiesto da VTK viene creato e gestito dalla classe Fl_ Gl Window appartenente all'insieme delle classi di FTLK.

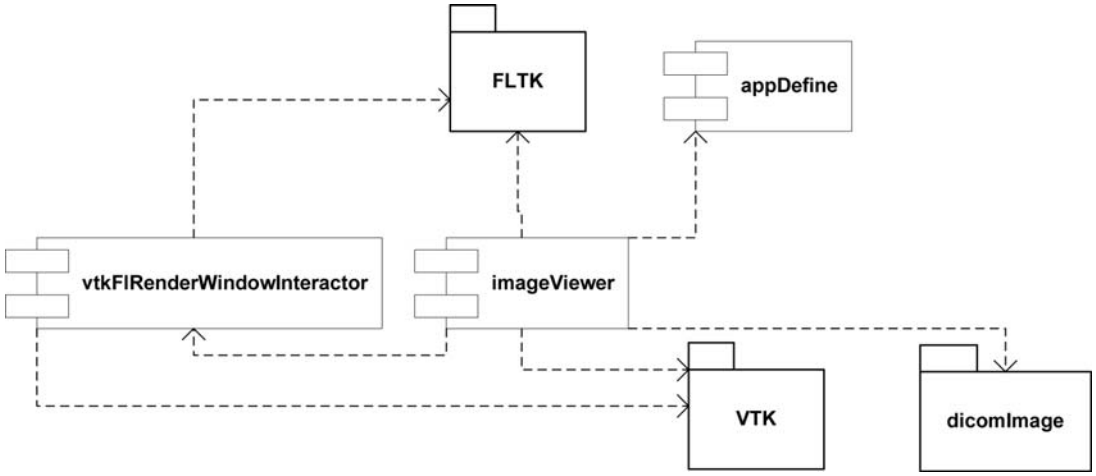

Figura 18 Component Diagram:imageViewer

L'oggetto imageViewer e' creato da un oggetto win2DGUI ed il metodo show, inizializza la pipeline di visualizzazione. Il metodo addDicomImage aggiunge alla lista degli oggetti dicomImage un nuovo oggetto e l'attributo p_dicomImage fa riferimento all'oggetto selezionato per la visualizzazione. La classe dicomImage e' quindi istanziata dalla classe win2D e le operazioni di delete sono gestite dall'imageViewer.
Le operazioni di visualizzazione delle immagini sono state realizzate mediante la classe vtkImageActor, tale classe, consente di realizzare un piano in uno spazio tridimensionale sul quale applicare l'immagine di un oggetto dicomImage come texture. Il metodo showDicomImage e' responsabile della realizzazione di parte della pipeline di VTK necessaria per la visualizzazione delle immagini e dei modelli 3D. E' in questo metodo infatti che e' stato realizzato il controllo per la visualizzazione 2D o 3D in funzione dell struttura dell'oggetto dicomImage.
Se l'attributo m_model3d di quest'ultimo oggetto fa riferimento ad un modello tridimensionale allora vengono offerti all'utente i controlli necessari per la visualizzazione 3D e viene aggiunto alla pipeline un oggetto di tipo vtkActor che rappresenta il modello 3D, altrimenti, i controlli sono vincolati ad una visualizzazione bidimensionale.
In figura 19 viene mostrata un'immagine DICOM ruotata e scalata, mentre in figura 20 e'e mostrato un modello tridimensionale e la slice corrente.

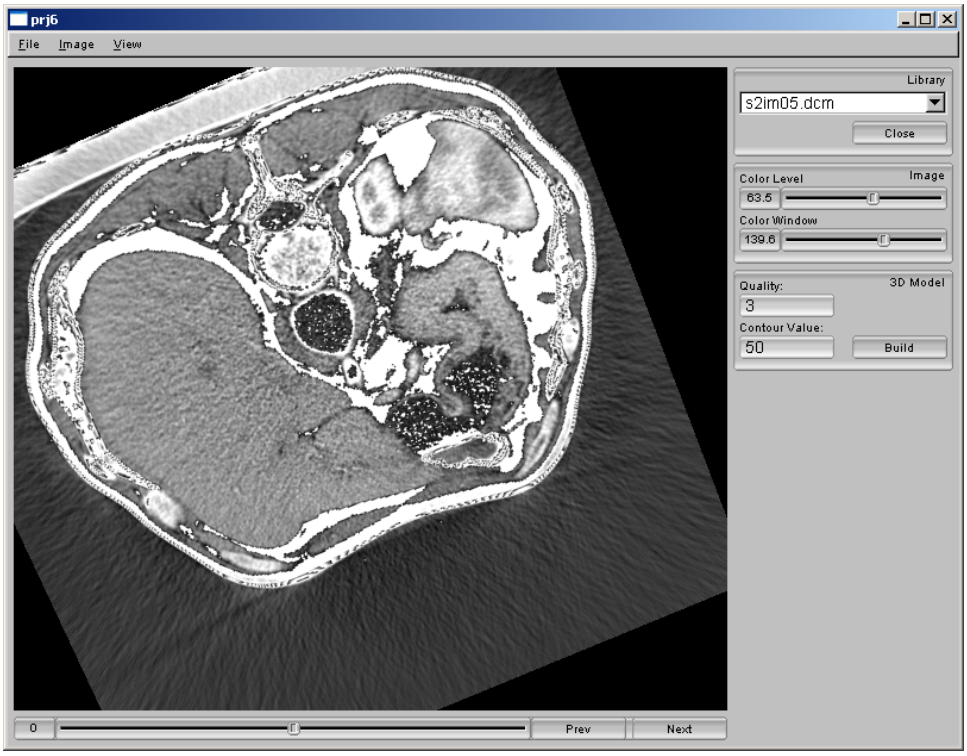

Figura 19 Interfaccia utente:visualizzazione slice

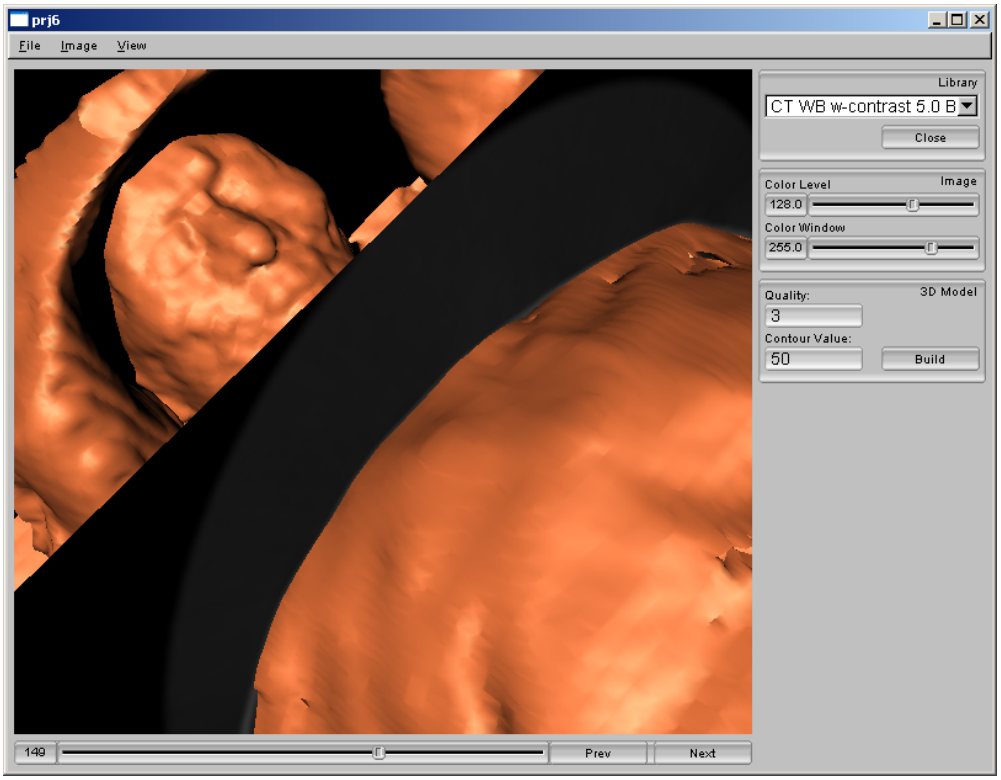

Figura 20 Visualizzazione modello 3d e slice

5 Conclusioni e sviluppi futuri

L'applicazione descritta in questo lavoro, presenta numerose caratteristiche essendo open source ed in particolare cross platform. Inoltre, essa e' stata progettata per offrire la massima flessibilita' verso l'inserimento di nuove caratteristiche e funzionalita' per la elaborazione e visualizzazione delle immagini di input.

Naturalmente, cio' non potra' essere fatto senza tenere nella giusta considerazione che l'evolversi delle tecnologie di acquisizione, il perfezionamento delle metodologie di scansione e di analisi dei fenomeni che riguardano gli organi umani, porteranno all'ampliarsi delle specifiche DICOM, sia in termini di risoluzione che di dinamica temporale, rendendo pertanto necessario lo studio e l'implementazione di nuove metodologie ed algoritmi di segmentazione e di registrazione.

La rappresentazione, inoltre, dovra' consentire forme sempre piu' immersive che consentano addirittura di spostarsi all'interno dell'organismo stesso, infatti il perfezionamento dei mezzi diagnostici, consentira' di discriminare piu' finemente i vari tessuti, con risoluzioni piu' elevate dal punto di vista dimensionale e di densita', il che permettera' di visualizzare, ad esempio il lume dei vasi sanguigni sempre piu' piccoli, per potersi spostare all'interno degli stessi e consentire la visione di placche di colesterolo, stenosi e cosi' via.

Attualmente e' gia' in fase di sviluppo un visualizzatore che consente una maggiore interattivita' con il modello tridimensionale mediante strumenti di stereoscopia, data glove, HMD (Head Mounted Displays), etc.. Il sistema descritto e' pronto per essere interfacciato con questo visualizzatore, grazie alla generazione di modelli OBJ, facilmente importati nell'ambiente di realta' virtuale.

.

Bibliografia

M. Kass, A. Witkin and D. Terzopoulos. "Snakes: Active contour models". Int. J. of Computer Vision, 2: 321-331, 1988.

J. Liberty's. "C++ Unleashed". SAMS, 1998.

W. E. Lorensen e H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". ACM Computer graphics, vol. 21, n. 24, pp. 163-169, luglio 1987.

Leszek A. Maciaszek. "Sviluppo di sistemi informativi con UML". Addison-Wesley, 1a edizione aprile 2002.

Morten Bro-Nielsen. "Medical image registration and surgery simulation". Ph.D. thesis at technical University of Denmark 1996.

B.H. McCormick, T.A. DeFanti and M.D. Brown. "Visualization in Scientific Computing". Report of the NSF Advizory Panel on Graphics, Image Processing and Workstations, 1987.

L. Rosenblum et al. *Scientific Visualization Advances and Challenges.* Harcourt Brace & Company, London, 1994.

J.T. Roff. "Fondamenti di UML". McGraw-Hill, 2003.

W. Schroeder, K. Martin and B. Lorensen."The Visualization Toolkit". *An Object-Oriented Approach to 3D Graphics.* Prentice Hall, 1998.

K. Waters and D. Terzopoulos. "Modeling and Animating Faces Using Scanned Data." *Visualization and Computer Animation.* 2:123-128, 1991.