Università degli Studi di Padova

Facoltà di ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Rilevazione dei movimenti indesiderati e casuali in immagini RMI

Laureando

Andrea Tomassetti

Relatore

Prof. Carlo Ferrari

Co-relatore

Prof. Diego Cecchin

Anno Accademico 2015/2016

"The best portion of a good man's life: his little, nameless unremembered acts of kindness and love."

William Wordsworth

Indice

Sommario

Lo studio delle immagini provenienti da risonanza magnetica nucleare è una delle principali tecniche di diagnostica per l'individuazione di masse tumorali o per indagini di quantificazione. Purtroppo, però, queste immagini non presentano grande risoluzione e anche piccoli movimenti possono creare zone d'ombra, artefatti, di cui, in fase di diagnosi, è difficile discernerne la natura. Un artefatto, ad esempio, potrebbe venire scambiato per una massa tumorale o, viceversa, potrebbe nasconderne una o non permettere di valutarne al meglio le dimensioni e la posizione. Avere informazioni chiare, precise e puntuali, mai come in questo caso, è di vitale importanza. Scopo di questa tesi è, pertanto, l'individuazione del movimento all'interno di questi studi. Grazie al programma realizzato il medico sarà in grado di scartare i dati che sono stati compromessi dal movimento del paziente. Per giungere a tale risultato le immagini vengono dapprima filtrare mediante l'applicazione di un filtro di Buttleworth, successivamente viene segmentato il contorno del cranio e, mediante l'utilizzo dei momenti, viene calcolato l'angolo di inclinazione. L'algoritmo risulta essere efficiente, da un punto di vista computazionale, ed efficace nell'individuazione dei movimenti. Anche se l'accuratezza nella stima dell'angolo di inclinazione non permette una ricostruzione del movimento.

Capitolo 1

Introduzione

La Positron Emission Topography/Magnetic Resonance (PET/MR), come verrà spiegato in maniera più esaustiva, è una nuova tecnologia che fonde i benefici di due tecnologie esistenti e maggiormente diffuse: la Positron Emission Topography (PET) e la Magnetic Resonance (MR). L'utilizzo principale di questa apparecchiatura risiede, tra gli altri, nello studio delle demenze e delle epilessie. Risulta quindi abbastanza evidente la necessità, data la natura dei pazienti, di avere uno strumento in grado di riconoscere il movimento involontario a cui questi soggetti sono inclini durante l'esame. Durante una risonanza magnetica, infatti, intervengono diversi tipi di disturbo, più o meno predicibili e correggibili, come: disomogeneità del campo magnetico dovuto alla presenza di oggetti metallici nel paziente, malfunzionamenti del circuito di rilevazione della radiofrequenza, il movimento del paziente. Tali tipi di disturbo provocano, nell'immagine finale, un artefatto che può essere definito come una caratteristica che appare nell'immagine ma che non è presente nell'oggetto esaminato. [1] Alcuni esempi di artefatti dovuti a movimento sono presentati in Figura 1.1.

L'origine degli artefatti da movimento può avere natura diversa: volontaria o involontaria. Sono considerati movimenti di natura involontaria il pulsare di un'arteria, il battere del cuore o il respiro del paziente. In questi casi, per evitare artefatti, è sufficiente sincronizzare l'acquisizione di codifica di fase con l'evento che genera il movimento. Molti costruttori di apparecchi di risonanza magnetica hanno sviluppato algoritmi proprietari per rimuovere tali artefatti. Alcuni nomi di sequenze progettate per rimuovere gli artefatti da movimento respiratorio sono, ad esempio, *respiratory gating*, *respiratory compensation* e *respiratory triggering*. Un esempio di applicazione di una se-

quenza per l'eliminazione del movimento respiratorio e circolatorio è visibile in Figura 1.2. Per quanto riguarda, invece, gli artefatti da movimento volontario non esiste ancora alcun tipo di tecnologia, proprietaria o meno, che riesca ad individuarli e correggerli efficacemente. Obbiettivo di questa tesi sarà, dunque, l'individuazione e la correzione di questi movimenti involontari.

Questo testo percorrerà tutte le fasi del lavoro svolto per ottenere un applicativo in grado di individuare e correggere i movimenti involontari. Nel primo capitolo verranno presentati gli strumenti utilizzati e delle nozioni di base per meglio comprendere le scelte adottate successivamente. Nel secondo capitolo, invece, verrà presentato, passo passo, l'approccio che ha portato alla soluzione adottata per il problema e che verrà presentata nel terzo capitolo. I capitoli successivi racchiuderanno i risultati ottenuti e le conclusioni.

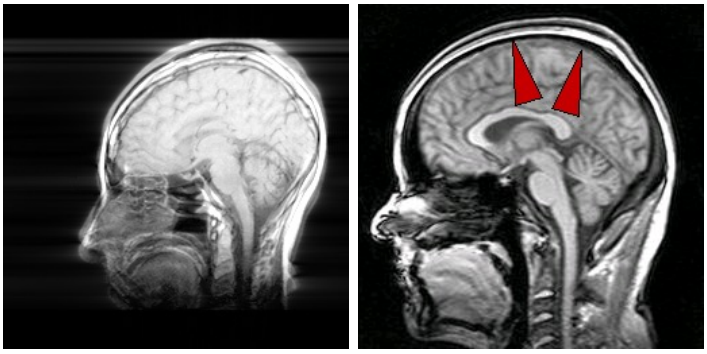
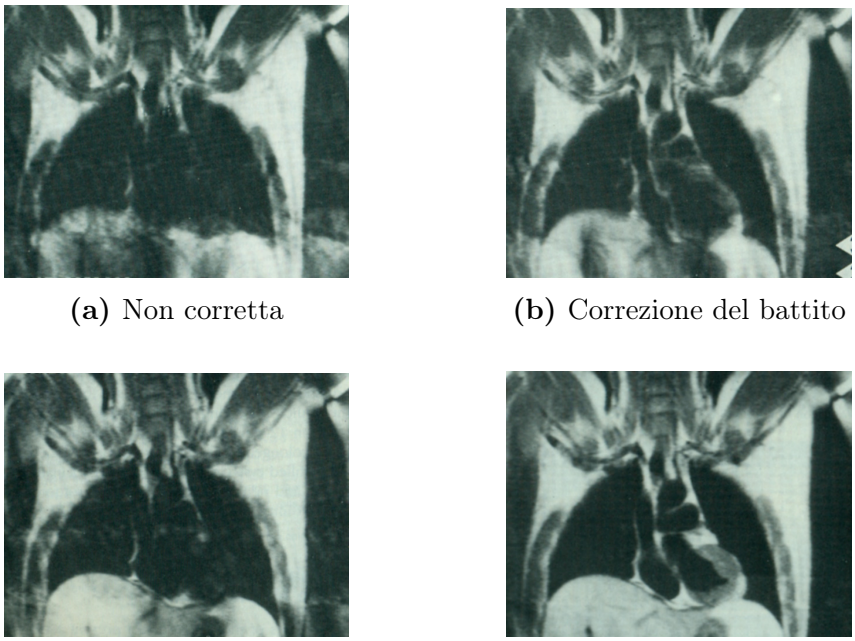


Figura 1.1: Esempi di artefatti causati dal movimento

1.1 PET/MR

Lo strumento, su cui è stato basato lo studio di questa tesi, è una PET/MR. I primi due esemplari di tale apparecchiatura sono di recente costruzione, essi infatti furono installati da *Philips* nel poi non così lontano 2010. Il primo, dei due esemplari, fu montato presso il *Mount Sinai Medical Centre*, negli Stati Uniti mentre il secondo si trova presso il *Geneva University Hospital*, in Europa. Questi primi modelli, però, erano da considerarsi alla stregua di veri e propri prototipi. Bisognerà infatti aspettare il 2011 prima che alla *Siemens* vengano concesse le autorizzazioni necessarie alla commercializzazione del prodotto finito, che comincerà ad essere venduto non prima del 2014. Il 26 Gennaio 2015, mediante un comunicato stampa, l'Azienda Ospedaliera di Padova comunica l'acquisto di una PET/MR della Siemens. Grazie a questo



(a) Non corretta **(b)** Correzione del battito

(c) Correzione della respirazione **(d)** Entrambe le correzioni

Figura 1.2: Correzione del movimento dovuto alla respirazione e al battito del cuore

acquisto, l'Azienda Ospedaliera è la prima in Italia ad aggiudicarsi una tale apparecchiatura.

L'utilizzo di questa nuova tecnologia, che ne combina insieme due già esistenti, permette di ridurre di un terzo l'esposizione dei pazienti alla radiazioni ionizzanti ricorrendo all'utilizzo di campi magnetici al posto dei Raggi X delle più tradizionali Positron Emission Topography/Computed Tomography (PET/CT). Per questo è particolarmente indicata per lo studio delle patologie oncologiche dei bambini. Questo strumento offre, con un'unica indagine diagnostica, le informazioni fornite dalla PET, ovvero quelle sulla funzionalità e vitalità cellulare, e quelle della MR, che fornisce invece informazioni relative alla morfologia e caratterizzazione tessutale. Questa nuova modalità *ibrida* di diagnostica d'immagini rappresenta la frontiera più avanzata del *molecular imaging*. Inoltre, altri vantaggi del Magnetic Resonance Imaging (MRI) rispetto all'*imaging* tradizionale sono: l'utilizzo di radiazioni non ionizzanti, la multiplanarità e soprattutto la multiparametricità. Quest'ultimo aspetto è legato ad una caratteristica intrinseca delle metodiche Nuclear Magnetic Resonance (NMR) dove si perturba il sistema di interesse rispetto alla posizione di equilibrio e si misura il segnale emesso dagli *spin* protonici

durante la successiva fase di rilassamento. [2]

La PET/MR risulta particolarmente utile per la diagnosi e la stadiazione delle neoplasie, in particolare quelle cerebrali, del collo, della mammella, del fegato e dell'apparato uro-genitale, nonché per lo studio delle patologie del muscolo cardiaco ed in campo neurologico, specie per lo studio delle demenze e delle epilessie. L'apparecchiatura, proprio per le sue peculiari caratteristiche, è destinata a svolgere prevalentemente un ruolo di ricerca clinica con il coinvolgimento diretto di specialisti in Medicina Nucleare, Radiologie e Neuroradiologia, in stretta collaborazione con i clinici. I principali campi di interesse, comprensivi delle patologie dell'adulto e del bambino sono oncologia, neurologia e cardiologia.

Durante la stesura di questo elaborato verranno utilizzati, come sinonimi, gli acronimi: MR, MRI e NMR. Storicamente, furono assegnati una sostanziosa varietà di nomi ed acronimi al processo di misurazione dell'assorbimento e dell'emissione di energia da parte di nuclei sottoposti all'azione di un campo magnetico. Nella letteratura fisica degli anni '40, tale fenomeno fu chiamato *nuclear induction* (induzione nucleare); nei primi anni '50, il suo nome fu modificato in *nuclear paramagnetic resonance* (risonanza paramagnetica nucleare). Dalla fine degli anni '50, però, il termine NMR fu di gran lunga il più utilizzato e il preferito per descrivere questo processo fisico. Quando si capì che questo metodo poteva essere utilizzato per scopi di diagnostica medica, venne dapprima utilizzato il termine NMR *imaging*. Successivamente, a metà degli anni '80, probabilmente a causa della preoccupazione suscitata nei pazienti dal termine *nuclear*, tale acronimo fu largamente dismesso in favore di MR o MRI. Al giorno d'oggi, il termine NMR è da preferire quando bisogna descrivere il processo fisico di per sé. Viene fatto decadere il termine *nucleare* quando ci si riferisce, invece, alle tecniche di *imaging* e spettroscopia su esseri umani e animali.

1.2 DICOM

Fin dalla sua prima pubblicazione, nel 1993, Digital Imaging and Communications in Medicine (DICOM) ha rivoluzionato il mondo della radiologia e dei *device* medici più in generale, permettendo di rimpiazzare le pellicole impresse dai raggi X con una loro controparte completamente digitale. DICOM è lo standard *de facto* delle immagini di natura medica e delle informazioni ad esse collegate (ISO 12052). Esso definisce il formato tramite il quale le

immagini mediche possono essere interscambiate con la qualità e la robustezza necessaria ad un loro corretto utilizzo clinico. Lo standard DICOM viene oramai implementato in quasi tutti i *device* medici che devono eseguire operazioni di *imaging*.

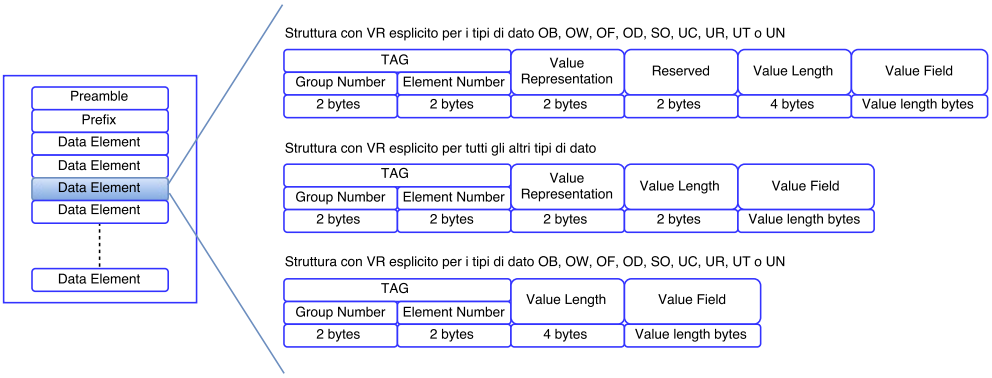


Figura 1.3: Struttura di un file DICOM

Ogni file DICOM è strutturato come riportato in Figura 1.3. Questi file contengono informazioni di natura diversa al loro interno, come ad esempio, dati relativi al paziente, dati di calibrazione del sistema di *imaging* utilizzato e i dati grezzi dell'immagine prodotta da tale sistema. Analizzando la struttura di un file DICOM si può notare che esso è costituito da

Header esso è costituito da un *File Preamble* di 128`byte` seguito da un prefisso DICOM di 4`byte` di lunghezza. Nel dettaglio, dunque, l'*header* è costituito da

> **Il *Preamble***. Lo standard DICOM non definisce alcun tipo di struttura per il *Preamble* ma definisce solo che debba essere di lunghezza fissa. Esso è stato pensato per agevolare l'accesso all'immagine, o ad altri dati contenuti nel file stesso, mantenendo la compatibilità con diversi formati di immagine più comunemente utilizzati.
>
> **DICOM *prefix***. Per questo campo, invece, lo standard è molto rigido. I 4`byte` devono infatti contenere la stringa di caratteri "DICM" codificata con caratteri maiuscoli e secondo lo standard ISO 8859 G0 Character Repertoire.

Finito l'*header*, in modo contiguo, comincia un *data set*: esso dovrebbe essere unico per ogni file DICOM. Il *data set*, a sua volta, è strutturato in *data elements* o *attributes* ognuno dei quali deve poter essere identificato in maniera univoca mediante un *Data Element Tag*. Un *data element* può essere espresso tramite tre strutture differenti. Due di queste strutture contengono entrambe il Value Representation (VR) dell'attributo ma differiscono per come viene espressa la lunghezza dell'intero campo, mentre l'altra struttura possibile non contiene alcuna informazione circa la rappresentazione del dato. Tutte e tre le differenti tipologie di struttura sono rappresentate in Figura 1.3. All'interno di un file DICOM devono poter essere salvati dati di natura particolarmente eterogenea, ogni dato, inoltre, possiede una sua diversa unità di misura. Inizia a delinearsi il problema, dunque, di riuscire a salvare e successivamente recuperare questi dati senza alcuna perdita di informazione. Per risolvere tale inconveniente lo standard DICOM ha introdotto i VR. Essi altro non sono se non rappresentazioni di ogni possibile dato clinico mediante l'utilizzo di tipi di dati base, nello standard ne sono stati definiti 27. Ognuna di queste rappresentazioni è identificata da

a. una coppia di lettere, che ne definisce il tipo di dato rappresentato
b. una descrizione dei caratteri concessi
c. una lunghezza prefissata del dato

Alcuni VR sono riportati in Tabella 1.1.

Riassumendo, pertanto, ogni *dataset* presenta un elemento, o attributo, la cui struttura è composta da

TAG. Una TAG, espressa nella forma (XXXX,XXXX), che identifica univocamente l'elemento
VR. Il tipo di dato base, o VR, che descrive il tipo di dato e il formato del valore dell'attributo
Value Length. La lunghezza del valore contenuto nel campo *value*
Value Field. Campo contenente i dati dell'attributo

Ai fini di questa tesi si è reso necessario l'utilizzo di un sottoinsieme di attributi presenti nei file DICOM. Essi verranno ora presentati e ne verrà descritto il loro utilizzo:

Identificativo	Tipo di dato
SH	Short String
LT	Long Text
UT	Unlimited Text
US	Unsigned Short
SL	Signed Long
FL	Floating point single
PN	Person Name
AS	Age String
UI	Unique Identifier
UN	Unknown

Tabella 1.1: Esempio di alcuni dati base definiti dallo standard

Image position (0020,0032) contiene il valore delle coordinate x, y e z dell'angolo in alto a sinistra dell'immagine. Tale dato viene utilizzato per determinare la posizione del paziente e, quindi, decidere se le immagini, prima di essere processate, devono essere sottoposte a rotazione (solitamente di 180°)

Image index (0054,1330) questo valore rappresenta la posizione dell'immagine all'interno di una serie PET

Number of slices (0054,0081) il numero di *slice*, assieme al valore precedente, viene utilizzato, durante la fase di pre-processing, per organizzare la serie di immagini[1]

[1]Di questo procedimento verranno forniti maggiori dettagli nei capitoli a seguire

1.3 OpenCV

Figura 1.4: OpenCV Logo

Open source Computer Vision library (OpenCV), come dice il nome stesso, è una libreria *open source* liberamente scaricabile dal sito di riferimento del progetto (**opencv.org**). La libreria è stata scritta in C e C++ ed è compatibile con qualsiasi ambiente di lavoro, che sia esso Linux, Windows, Mac OS X, iOS o Android. Al suo interno comprende più di 2500 algoritmi, tra i quali vi è anche un'esaustiva raccolta di algoritmi di *computer vision* e *machine learning* che rappresentano lo stato dell'arte. La libreria, inoltre, mette a disposizione dei programmatori una serie di Application Programming Interface (API) disponibili per diversi linguaggi di programmazione come, ad esempio, Python, Java, Ruby, Matlab e altri. I creatori di OpenCV hanno creato tale libreria con un occhio di riguardo alle performance e all'efficienza computazionale, ponendo grande attenzione ad un suo utilizzo in applicazioni *real-time*. Questo obbiettivo è stato perseguito operando ottimizzazioni a diversi livelli: dalla scrittura di algoritmi particolarmente efficienti, allo sfruttamento delle Central Process Unit (CPU) multicore fino allo scrivere direttamente parte del codice in assembly.[3] Date queste premesse, dovrebbe iniziare ad essere chiaro perché è stata scelta questa libreria per perseguire l'obbiettivo di questa tesi.

OpenCV affonda le sue origini nei laboratori di ricerca del gruppo Intel© che lanciò, negli anni, diversi progetti legati alla *computer vision*.

Uno degli autori di questi progetti a quel tempo si trovava in visita presso alcune università dove notò che alcuni gruppi di studenti delle università più facoltose, come il MIT, avevano sviluppato e rilasciato internamente del codice ben strutturato in grado di gestire applicazioni per la visione artificiale. Questo progetto interno veniva tramandato di studente in studente e, ad ogni passaggio, veniva ampliato e migliorato. Invece di reinventare la ruota e riscrivere tutte le funzioni base dal nulla, Intel decise di utilizzare quel progetto come base della libreria OpenCV.

Con il termine *computer vision* si intende la trasformazione di immagini statiche 2D/3D in una decisione o, comunque, in una nuova rappresentazione. Tutte queste trasformazioni perseguono, chiaramente, il fine ultimo di raggiungere un prefissato obbiettivo. La sfida vera e propria della *computer vision* è quella di estrapolare informazioni sul mondo a partire dalle immagini, di riprodurre la vista umana da qui il nome italiano visione artificiale. Tra i diversi campi di utilizzo di questo processo tecnologico si trovano applicazioni anche in campo medico: le ultime e innovative scoperte nel campo della visione artificiale permettono di migliorare notevolmente la qualità della diagnosi e, quindi, della cura dei pazienti. [4]

A partire dalla versione 2.2, OpenCV risulta divisa in diversi moduli. Questi moduli vengono compilati insieme alla libreria ed ognuno espone delle funzioni differenti che comprono campi diversi della *computer vision*. I moduli sono i seguenti:

`opencv_core` è il modulo che contiene le funzionalità base della libreria, in particolar modo: le strutture e i tipi base e le funzioni aritmetiche

`opencv_imgproc` è il modulo che contiene le principali funzioni per il *processing* delle immagini

`opencv_highgui` è il modulo che contiene le funzioni per la lettura e salvataggio delle immagini e dei video insieme ad altre funzioni per interagire con l'utente attraverso un'interfaccia grafica

`opencv_features2d` è il modulo dedicato al *pattern matching*. Esso contiene, infatti, funzioni per individuare descrittori all'interno di un'immagine

`opencv_calib3d` è il modulo che contiene funzioni atte alla calibrazione di un sistema di camere e alla riscostruzione stereografica a partire da un'immagine presa da due punti di vista diversi

`opencv_video` è il modulo dedicato al video: contiene funzioni per il riconoscimento dello sfondo e del *foreground* nonché funzioni per il tracciamento degli oggetti in movimento

`opencv_objdetect` è il modulo che contiene le primitive per il riconoscimento degli oggetti come, ad esempio, il viso delle persone

Inoltre la libreria contiene altri moduli dedicati al *machine learning* (`opencv_ml`) e all'accelerazione grafica mediante Graphics Processing Unit (GPU) (`opencv_gpu`).

A tutti questi moduli è associato un *header*, che è possibile trovare all'interno della cartella `include`. Un file C/C++ classico comincia con l'inclusione dell'*header* dei moduli che verranno utilizzati, per esempio:

```
1 #include <opencv2/core/core.hpp>
  #include <opencv2/imgproc/imgproc.hpp>
3 #include <opencv2/highgui/highgui.hpp>
```

Nonostante la documentazione online sia più che esaustiva, viene riportata qui di seguito una breve guida passo passo per la compilazione della libreria.

Dal sito di OpenCV, o direttamente dal *repository git*, è possibile scaricare il codice sorgente della libreria. Una volta effettuata tale operazione ci si troverà davanti una serie di cartelle: la cartella `doc` contiene la documentazione di OpenCV, la directory `include` contiene tutti gli *header*, nella cartella `modules` è presente il codice sorgente mentre, infine, nella cartella `samples` sono contenuti degli esempi completi di codice pronti all'uso.

Prima di poter utilizzare la libreria essa deve essere compilata, generandone così i binari, utilizzando l'appropriato compilatore C++. Per generare i *makefiles* necessari alla compilazione della libreria si può usare il tool CMake (**cmake.org**), il cui funzionamento è illustrato in Figura 1.5. CMake è uno strumento open source progettato per controllare il processo di compilazione di applicativi e librerie mediante l'utilizzo di file di configurazione indipendenti dalla piattaforma su cui si trova ad essere eseguito. Dopo aver avviato il programma `cmake-gui` e aver selezionato la cartella dove si trovano i sorgenti appena scaricati di OpenCV e la cartella dove si vuole che essi vengano compilati, si può procedere a premere il tasto `Configure`. Una volta finita la fase di configurazione ci si troverà davanti ad una schermata

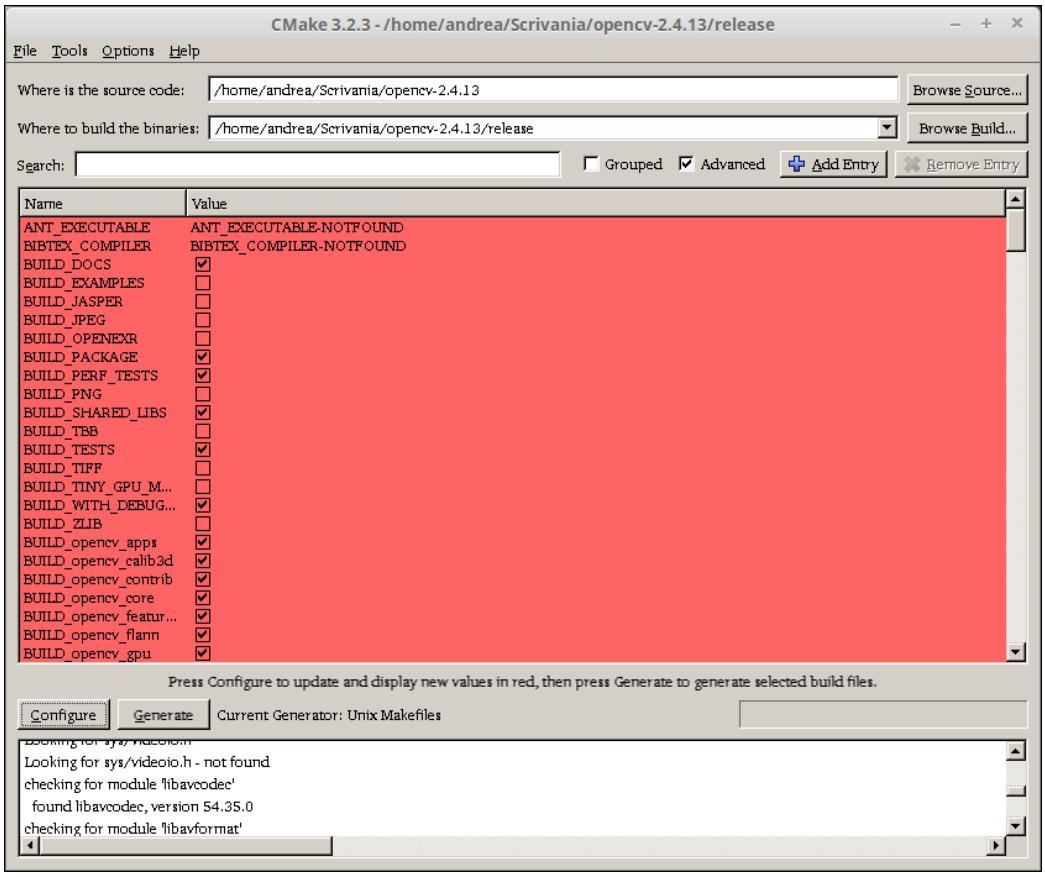


Figura 1.5: CMake e OpenCV

come quella riportata in Figura 1.5. [5] A questo punto si può scegliere, a seconda delle necessità, quali moduli di OpenCV compilare, che ottimizzazioni abilitare, per che linguaggi di programmazione generare le API e molto altro ancora. Una volta selezionata la configurazione ottimale si può procedere alla compilazione vera e propria, per fare ciò basterà premere il pulsante `Generate` e, successivamente, aprire un terminale nella cartella di output e lanciare i comandi

```
$ make
# make install
```

Capitolo 2

Approccio

Nelle sezioni di questo capitolo verranno affrontate le tematiche relative all'approccio che ha permesso di giungere ad una soluzione del problema in esame. Inizialmente, verranno illustrate e analizzate nel dettaglio le problematiche di tale studio: verranno date le risposte ad alcune domande, come, perché si sta cercando di affrontare questi quesiti, come è stato affrontato nella letteratura e qual è l'idea innovativa alla base di questo approccio. Successivamente verrà posta particolare attenzione al dataset, all'insieme di dei dati, su cui si è basato il lavoro di questa tesi. Si potranno così intuire parte delle difficoltà incontrate. Infine, verranno illustrate le strade intraprese, gli errori commessi, i vicoli ciechi imboccati.

2.1 Analisi del problema

Nel momento in cui le immagini mediche devono venire utilizzate per scopi diagnostici risulta fondamentale, o, se non altro, fortemente auspicabile che esse siano il più fedeli possibile alla realtà. Ciò purtroppo, come accennato nel capitolo precedente, non è un vincolo che si riesce sempre a rispettare. Oltre tutti i tipi di disturbo e di rumore generati durante l'acquisizione, il movimento del paziente è uno dei fattori predominanti e più distruttivi in termine di informazione dell'immagine. Anche piccoli movimenti possono creare zone d'ombra, artefatti, di cui, in fase di diagnosi, è difficile discernerne la natura. Un artefatto, ad esempio, potrebbe venire scambiato per una massa tumorale o, viceversa, potrebbe nasconderne una o non permettere di

valutarne al meglio le dimensioni e la posizione. Avere informazioni chiare, precise e puntuali, mai come in questo caso, è di vitale importanza.

In questo scenario si colloca il lavoro di questa tesi. Tale lavoro è stato coadiuvato, nonché sponsorizzato e in parte commissionato, dal professor Diego Cecchin, professore associato presso il Dipartimento di Medicina (DIMED) dell'Università degli Studi di Padova. Il prof. Cecchin si occupa ormai da diversi anni di diagnostica per immagini mediche diventando, di fatto, uno dei maggiori esperti nel campo della medicina nucleare, campo, per altro, nel quale ha condotto i suoi studi di specializzazione. Da lui nasce la volontà di migliorare la qualità del contenuto informativo delle immagini prodotte dalla NMR, nessuno dei tool disponibili attualmente, a suo parere, è in grado di correggere efficacemente i movimenti dei pazienti. Nel suo caso specifico, inoltre, i movimenti sono particolarmente presenti e significativi: il suo campo di ricerca prevede, infatti, lavorare con pazienti che soffrono di Alzheimer o di demenza. Essi, quindi, nonostante la collaborazione e la determinazione sono naturalmente impossibilitati a mantenere una condizione di stabilità statica. Se si aggiunge poi il fatto che l'acquisizione dura circa un'ora, risulta facile intuire che saranno presenti, alla fine, degli artefatti da movimento.

Dall'esperienza, maturata negli anni dal prof. Cecchin nel settore dell'*imaging* medico, è emersa la mancanza di un tool valido per il rilevamento e la successiva correzione del movimento. Ad esempio, pmod© non si è rivelato all'altezza delle aspettative. Ulteriore difficoltà alla risoluzione del problema è introdotta dal tipo di immagini in esame: esse, oltre a presentare un esiguo contenuto informativo, cambiano nel tempo senza alcun tipo di regolarità. Nonostante, infatti, il contrasto inerente MR possa essere manipolato in misura molto maggiore rispetto ad altre tecniche di *imaging*, alcune domande di diagnostica non possono essere risolte facilmente e richiedono l'applicazione di agenti di contrasto. [6] Un mezzo di contrasto viene solitamente somministrato per endovena, con l'eccezione dei mezzi di contrasto specifici per lo studio dell'apparato digerente. Dopo l'iniezione, il mezzo di contrasto viene veicolato dal sistema circolatorio attraverso il corpo. A questo punto i mezzi di contrasto vanno a distribuirsi a seconda della loro funzionalità. Un mezzo di contrasto è una sostanza che viene introdotta nel corpo per aumentare il contrasto tra i tessuti. I mezzi di contrasto sono principalmente costituiti da sostanze paramagnetiche, ma possono essere utilizzate anche sostanze ferromagnetiche. Con meccanismi di azione differenti questi due tipi di sostanze agiscono sui tempi di rilassamento dell'acqua.

La base per una ricerca della soluzione a questo problema è stata gettata a partire dallo studio della letteratura precedente. Come era stato affrontato questo problema in passato? Cercando tra la gli articoli riguardanti questa tematica si trovano diverse tipologie di approccio: Fulton, Meikle *et al.* nella loro pubblicazione, intitolata *Correction for Head Movements in Positron Emission Tomography Using an Optical Motion Tracking System* [7], dimostrano quanto sia accurato individuare e correggere i movimenti dei pazienti sottoposti a NMR mediante l'utilizzo di un sistema ottico di tracciamento del movimento della Polaris. Questo sistema, composto quindi da una telecamera esterna, fornisce i risultati sperati e riesce nell'impresa di correggere il movimento. Qualche anno più tardi, Olesen *et al.*, nel loro articolo intitolato *List-mode PET motion correction using markerless head tracking: proof-of-concept with scans of human subject* [8], dimostrano l'effettiva efficacia di questo sistema di tracciamento. In Figura 2.1 ne è riportata un'illustrazione. Il problema, chiaramente, rimane la necessità di fornirsi di un sistema di tracciamento, che deve essere calibrato per far si di mettere in relazione il sistema di riferimento della telecamera con quello della macchina di risonanza magnetica. Come si può facilmente intuire, è un metodo poco pratico, abbastanza complesso.

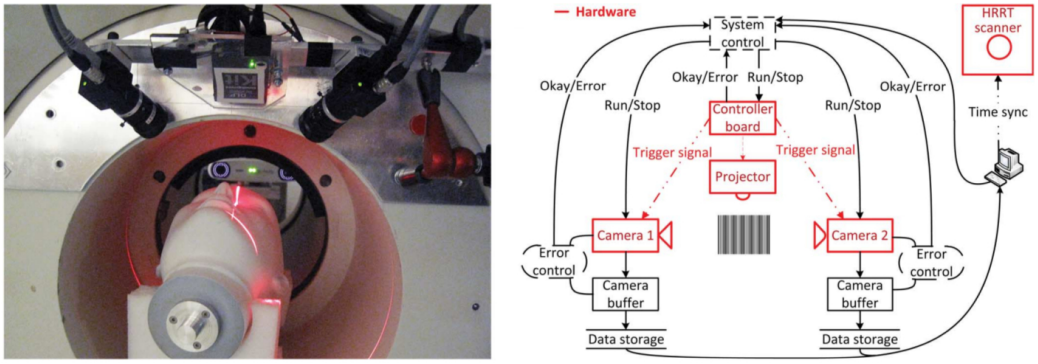


Figura 2.1: Sistema ottico di tracciamento del movimento per risonanza magnetica nucleare

Durante le fasi iniziali, quelle di approccio al problema, tra le altre proposte è stata vagliata anche quella di utilizzare dei marcatori da applicare sul volto del paziente. Attraverso l'utilizzo di questi *marker* sarebbe stato relativamente facile il compito di individuare gli spostamenti dovuti al movimento. Anche questo approccio sarebbe stato sicuramente invasivo ma in maniera significativamente minore rispetto a quello proposto da Fulton *et al.*. Purtroppo però l'utilizzo di marcatori, durante un esame eseguito con la macchina per risonanza magnetica nucleare, risulta essere una strada impra-

ticabile. Un marcatore, applicato sul paziente, per essere percepito e, quindi, essere infine presente nei dati elaborati dalla macchina deve contenere per forza del liquido di contrasto. Tale liquido è radioattivo e, oltre a tutte le complicazioni del caso, tende a deteriorarsi con il tempo.

Scartate, dunque, tutte le ipotesi che prevedevano l'utilizzo di strumentazione esterna, si è deciso di porre maggiore attenzione e focalizzare i ragionamenti su soluzioni che prevedessero i soli dati prodotti dalla NMR come sorgente di ingresso. Da tale sorgente si voleva realizzare un programma in grado di affermare se ci fosse stato o meno movimento e, se possibile, di correggerlo. Lo studio di fattibilità è stato condotto inizialmente sulla letteratura esistente. Nell'articolo *A survey of medical image registration* di Maintz e Viergever [9] vengono analizzate e citate le maggiori tecniche di *image resgistration* in ambito medico. Purtroppo il nostro obbiettivo è leggermente diverso, un'ottima definizione di tale procedura viene riportata nell'articolo di Kostelec e Periaswamy [10]: *registrare* due immagini significa *allinearle* in modo tale che le i punti in comune si sovrappongano mentre le differenze, sempre che ve ne siano, siano evidenziate e ben visibili anche ad occhio nudo. Alcuni esempi, in campo medico, in cui questa tecnica è di particolare aiuto sono: voler confrontare due Computed Tomography (CT) di uno stesso paziente prese una sei mesi a distanza dall'altra oppure allineare i dati provenienti da una *pet* con quelli provenienti da una MR (pratica utile quando non si dispone di una macchina che acquisisce entrambi i tipi di dato simultaneamente).

Nonostante la registrazione di immagini non coincida con il campo di ricerca di questo progetto, l'articolo di Maintz e Viergever è stato di grande ispirazione e ha permesso di valutare alcuni metodi ben collaudati per la soluzione di problemi molto simili a quello che si sta per andare ad affrontare. Nella quasi totalità degli approcci illustrati nell'indagine condotta dai due studiosi, per raggiungere lo scopo della registrazione delle immagini mediche viene considerata l'intera superficie a disposizione, l'intera figura. Pochissimi algoritmi infatti, principalmente quelli descritti da Pelizzari *et al.* in [11], [12] e [13], ricorrono alla segmentazione dell'immagine prima di applicare la registrazione della stessa. Il problema, però, è che la soluzione adottata negli articoli citati è solo in parte automatizzata. Mentre la registrazione, infatti, avviene in maniera totalmente automatica, lo stesso non si può dire per la segmentazione che ha, invece, bisogno di un intervento umano.

Data la natura mutevole delle immagini e l'enorme complessità del problema, si è cercato un approccio alternativo a quelli riportati nella let-

teratura. L'idea innovativa, alla base della soluzione sviluppata, consiste in una segmentazione del tutto automatica delle ossa che costituiscono il cranio. Queste, infatti, sono l'unica parte dell'immagine che rimane inalterata nel tempo, l'intensità non risulta infatti essere funzione del tempo perché esse non sono percorse da alcun mezzo di contrasto. Esse, inoltre, costituiscono un contorno sufficientemente definito tramite dal quale è possibile ricavare l'orientazione, successivamente verrà spiegato nel dettaglio il procedimento completo. Solitamente, quando si esegue un esame diagnostico di risonanza magnetica, ciò che si vuole analizzare sono i tessuti molli, non le ossa. Per questo motivo le immagini vengono sottoposte a una tecnica di *post-processing* chiamata correzione dell'attenuazione. In Figura 2.2 è riportato un esempio di procedimento.

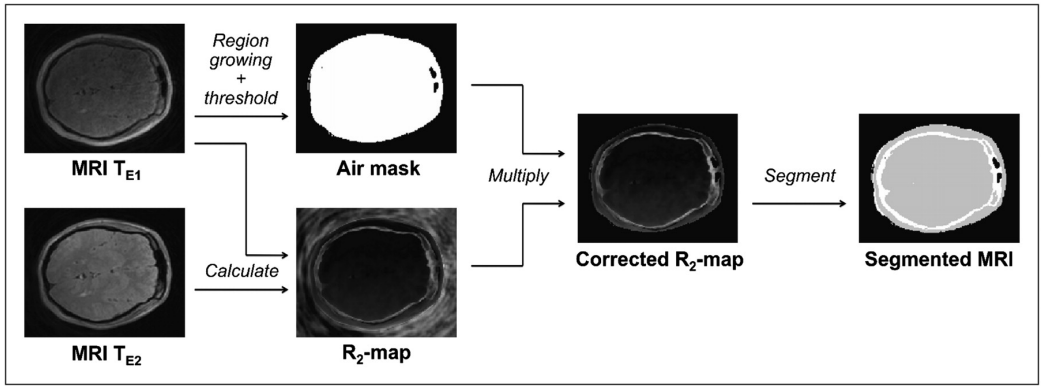


Figura 2.2: Funzionamento del processo di correzione dell'attenuazione

Come anticipato nel capitolo precedente, il segnale che viene misurato è quello emesso dagli *spin* protonici durante la fase di rilassamento. Tale segnale risulta dipendere da più parametri come la densità protonica, i tempi di rilassamento (T_1, T_2) e la *diffusione*. Il contrasto dell'immagine finale che si ottiene è espressione di tali parametri. La possibilità di pesare maggiormente un'immagine rispetto a ciascuno dei singoli parametri funzionali consente di ottenere immagini con un'ampia gamma di contrasto. [14] La diffusione, quindi, gioca un ruolo importante nel contrasto che assumerà l'immagine risultante prodotta dalla macchina. La diffusione è un processo tridimensionale in cui la mobilità molecolare non può essere la stessa in ogni direzione. Questa anisotropia può essere dovuta a vari fattori, tra cui la presenza di ostacoli e la configurazione strutturale del mezzo. Nei tessuti biologici, data la presenza di ostacoli quali fibre, macromolecole ed organelli, le molecole che diffondono percorrono sempre un cammino maggiore dell'effettivo spostamento: le molecole non possono, infatti, diffondere in linea retta,

ma intorno alle strutture per loro impermeabili. Tutto questo si traduce in un maggior tempo di diffusione per diffondersi da una regione ad un'altra. In Figura 2.3 è riportata una rappresentazione grafica del concetto appena espresso.

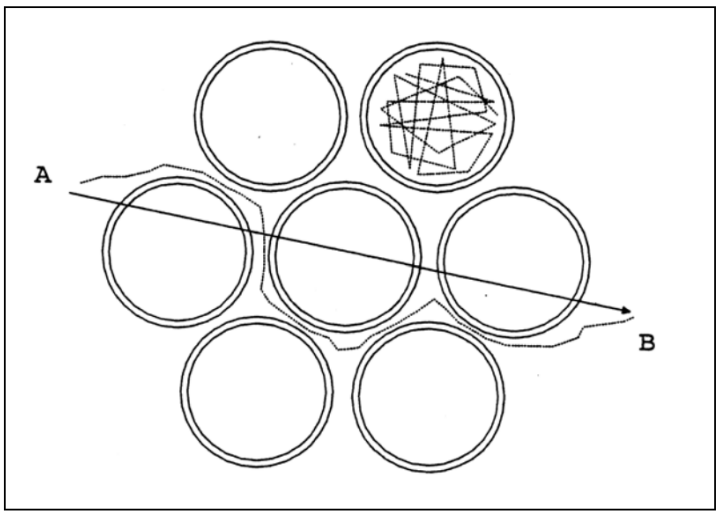
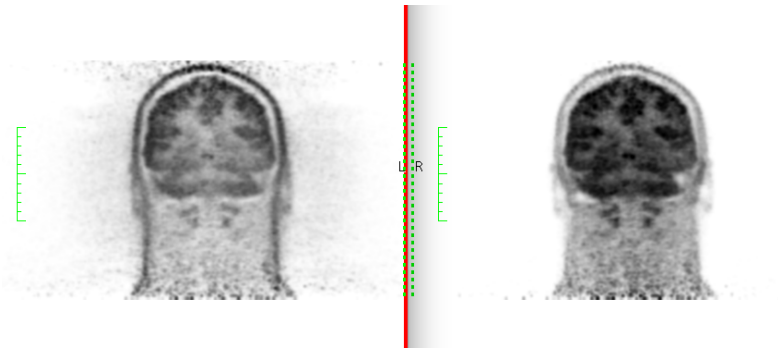


Figura 2.3: Diffusione ristretta (compartimento in alto) e diffusione ostacolata. Le varie strutture all'interno dei tessuti impediscono alle molecole di seguire un percorso rettilineo da A a B, costringendole ad effettuare un cammino tortuoso.

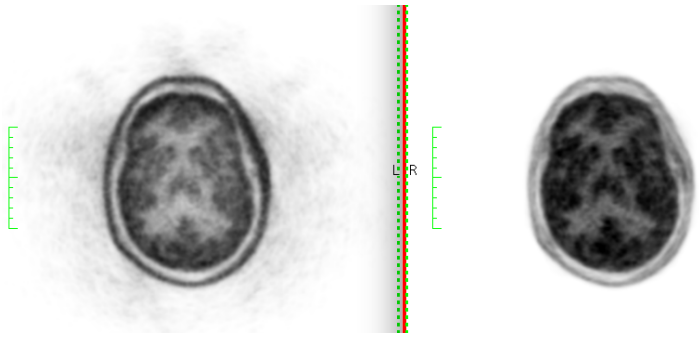
Il processo di attenuazione della correzione, dunque, ha come obbiettivo quello di evidenziare le zone centrali, quelle di maggior interesse, ma che, a causa del ritardo nella diffusione, risultano anche essere quelle con intensità minore. In questo modo, però, si perdono informazioni sui tessuti esterni, tra i quali le ossa che compongono il cranio, per questo motivo il lavoro svolto prenderà in esame le immagini senza correzione dell'attenuazione. In Figura 2.4 è illustrata la differenza tra immagini con attenuazione della correzione e senza.

2.2 Dataset

Il dataset, utilizzato per portare a compimento l'incarico assegnato, è stato fornito dal professor Cecchin. Si è pertanto lavorato su casi di pazienti reali, con tutte le problematiche che ciò può comportare. Una tra tutte: l'elevata rumorosità delle immagini. Quella mostrata in Figura 2.4 (a) ne è un esempio. Essa, nonostante sia stata ritagliata a fini tipografici, presenta un'elevata



(a) Vista coronale

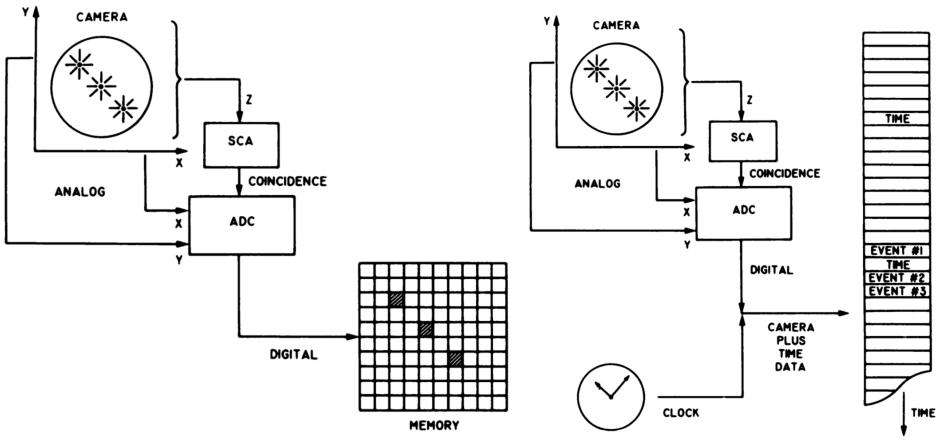


(b) Vista assiale

Figura 2.4: Comparazione di immagini provenienti da MR: nelle immagini a sinistra non è stata applicata la correzione dell'attenuazione mentre le immagini di destra risultano filtrate

quantità di rumore che caratterizza tutte le acquisizioni e, di conseguenza, tutti i dati presenti nel dataset. Trattandosi di pazienti reali, inoltre, per rispettare la loro privacy le loro informazioni personali sono state eliminate; anonimizzando così, di fatto, i dati.

Un'ulteriore considerazione da fare a proposito dei dati presenti nel dataset riguarda il loro metodo di acquisizione. Mentre un esame diagnostico classico genera un unico *frame*, per studi di quantificazione si ricorre ad una tecnica di acquisizione denominata *list-mode*. In questa modalità di acquisizione, la macchina acquisisce eventi lungo la linea temporale e, successivamente, si può decidere come ricostruirli, con che intervallo temporale e, quindi, con che risoluzione. Nel caso di acquisizione in *frame mode*, invece, questa ricostruzione a posteriori non è possibile. Infatti, come si può vedere da Figura 2.5a ogni evento rilevato processa i segnali analogici x, y e z provenienti dalla camera a scintillazione. Se il segnale ricade all'interno della finestra di energia, l'Analog to Digital Converter (ADC) digitalizza le



(a) Frame mode **(b)** List mode

Figura 2.5: Modalità di acquisizione a confronto

coordinate spaziali x e y del segnale e ne produce una discretizzazione che viene poi utilizzata per fare riferimento ad un indirizzo di memoria, sotto forma di *word* o *byte*, del PC. Tale indirizzo di memoria conterrà, quindi, un elemento dell'immagine altre sì chiamato *pixel*. In questo modo, dunque, un'intera acquisizione si riduce ad un'unica immagine, ad un unico *frame*. Il metodo di acquisizione in *list-mode*, invece, non genera un'immagine digitale bensì una lista di indirizzi in memoria che contengono, in ordine cronologico crescente, una serie di eventi, in cui vengono registrate le coordinate x e y del segnale, intervallati da dei *marker* temporali, detti *timestamp*. Questo comportamento è visibile in Figura 2.5b. Questa lista di eventi può poi essere raggruppata in *frame*, in questo modo si possono ottenere delle immagini su cui eseguire le dovute analisi. [15] L'utilizzo dell'acquisizione in *list-mode*, dunque, è largamente diffuso quando si conducono studi di quantificazione ed è molto utile anche per la ricostruzione del movimento. Infatti, Koss *et al.*, nell'articolo intitolato *Advantages Of List-Mode Acquisition Of Dynamic Cardiac Data*, affermano che "probabilmente l'attributo di maggior rilievo dell'acquisizione in *list-mode* è la possibilità di correggere il movimento del paziente". [16]

Come è già stato accennato, il dataset utilizzato per questo progetto deriva da casi reali. Non c'è, quindi, alcun controllo o alcuna informazione a priori su possibili movimenti o anomalie in generale. Per questo motivo si è scelto di modificare manualmente, a posteriori, alcuni di questi casi reali inserendo dei movimenti ad-hoc. Questi movimenti sono stati introdotti

per testare il corretto funzionamento e, più in generale, il comportamento del software sviluppato. Maggiori dettagli sui risultati prodotti verranno presentati nei capitoli successivi e, in modo particolare, nel Capitolo 4.

2.3 Strade intraprese

Durante le primissime fasi di approccio alla soluzione del problema sono state vagliate diverse strade. Innanzitutto si è cercato un metodo efficace per segmentare ed isolare la parte di interesse dell'immagine: il cranio. Le immagini prodotte dalla risonanza magnetica possono essere di diversi formati, di differenti risoluzioni. Per lo svolgimento di questo lavoro ci si è focalizzati su immagini provenienti da volumi $344 \times 344 \times 127$, di cui l'area di effettivo interesse ricopre solo 100×100 pixel. Le immagini, inoltre, presentano una forte componente rumorosa che deve essere eliminata prima di poter iniziare il procedimento di segmentazione. Inizialmente, questa componente, veniva filtrata utilizzando un *box linear filter*. Un *box linear filter*, o *box blur*, altro non è se non un'approssimazione lineare di un filtro gaussiano. Esso applica, iterativamente all'intera immagine, un *kernel* composto da coefficienti identici, come riportato in 2.1. In questo modo, l'effetto finale, sarà quello di *smoothing*. I filtri di tipo gaussiano, inoltre, sono di inevitabile utilizzo quando si devono evidenziare i bordi. [17]

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.1}$$

Successivamente all'applicazione del filtro, si è deciso di eseguire una soglia. Il tipo di soglia scelto è stato quello di soglia a zero il cui funzionamento, riportato in 2.2, è molto semplice: tutti i pixel di intensità minore all'intensità di soglia vengono spenti, ovvero posti uguali a 0, mentre gli altri mantengono il loro valore.

$$\mathrm{dst}(x, y) = \begin{cases} \mathrm{src}(x, y) & \text{if } \mathrm{src}(x, y) > \mathrm{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Purtroppo però entrambe le tecniche adottate sono risultate inefficaci perché troppo semplicistiche per il problema in esame. Come si può osservare in Figura2.6, infatti, la componente rumorosa che, inizialmente, era predominante, è stata praticamente azzerata. Durante quest'operazione, però, anche alcune aree del cranio sono state erose lasciando dei salti, una linea spezzata dove prima, invece, c'era continuità. Inoltre il rumore residuo si concentra nelle vicinanze del bordo del cranio rendendone, di fatto, impossibile una corretta individuazione.

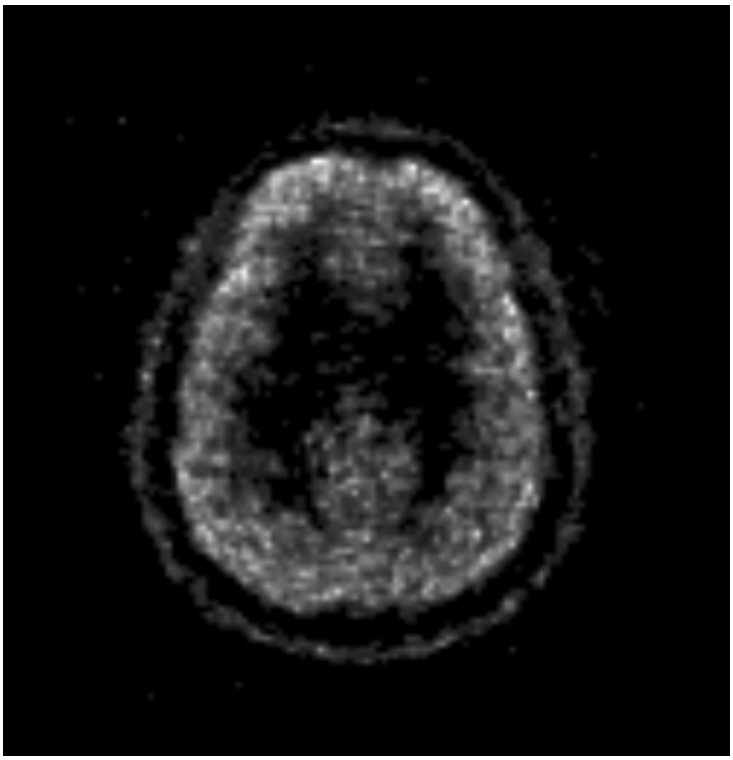


Figura 2.6: Risultato dell'applicazione di un *box filter* e di una soglia a zero

Per migliorare, soprattutto dal punto di vista dei contorni, un'immagine si ricorre spesso alle trasformazioni morfologiche. Con le trasformazioni morfologiche si modificano regioni di un immagine attraverso elementi strutturanti, matrici binarie che guidano l'operazione di trasformazione. Una delle applicazioni principali di queste trasformazioni è l'eliminazione di piccole caratteristiche spurie presenti in un'immagine binaria attraverso un processo chiamato erosione, seguito da un processo complementare chiamato dilatazione. Più in generale, questa operazione morfologica prende il nome di operazione di apertura. Entrando più nello specifico, la dilatazione aumenta l'ampiezza delle regioni di intensità maggiore in questo modo è in grado di

rimuovere piccole imperfezioni rumorose dall'immagine. La dilatazione di A attraverso l'elemento strutturante B è definita come

$$A \oplus B = \bigcup_{b \in B} A_b \tag{2.3}$$

L'operazione di erosione, invece, può essere considerata come l'operazione inversa a quella di dilatazione. Essa infatti è definita come

$$A \ominus B = \{z \in E | B_z \subseteq A\} \tag{2.4}$$

Dove E rappresenta uno spazio euclideo e A un'immagine binaria in esso contenuta. Mentre B_z rappresenta la traslazione di B, elemento strutturante, per il vettore z, ovvero $B_z = \{b + z | b \in B\}$, $\forall z \in E$. [18] Nemmeno l'applicazione di queste trasformazioni ha reso possibile ottenere delle immagini più pulite, in cui il contorno del cranio fosse facilmente individuabile e segmentabile.

In letteratura l'approccio computazionale più famoso, rinomato ed efficace per l'individuazione dei bordi all'interno di un'immagine è il metodo di Canny. [19] Il metodo prende il nome dall'autore dell'articolo. Lo scopo dell'algoritmo di *Canny* è quello di soddisfare tre principali requisiti:

1. Basso tasso di errore, ovvero individuare solo bordi realmente esistenti

2. Buona localizzazione, ovvero rendere il più possibile nulla la distanza, in pixel, tra il bordo individuato e quello reale

3. Risposta minima, cioè una sola risposta per ogni bordo

Per arrivare a tale obbiettivo il filtro di Canny compie diversi passi:

1. Filtra il rumore. Per questo scopo viene utilizzato un filtro di Gauss, solitamente con *kernel* 5 come quello riportato in 2.5

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.5}$$

2. Successivamente calcola il gradiente dell'intensità dell'immagine. Per fare ciò applica una coppia di maschere di convoluzione, nelle direzioni x e y

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{2.6}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2.7}$$

Infine ricava il module e l'angolo del gradiente utilizzando:

$$\begin{aligned} G &= \sqrt{G_x^2 + G_x^2} \\ \theta &= \arctan(\frac{G_x}{G_y}) \end{aligned} \tag{2.8}$$

L'angolo risultante viene in seguito approssimato con uno tra 0, 45, 90, 135.

3. Rimuove i punti *non-massimi*. In questo modo rimangono solo bordi candidati sottili perché i pixel che non sono considerati parte di un bordo vengono rimossi.

4. Infine, applica l'ultimo passaggio: l'isteresi. Per fare questo *Canny* utilizza due soglie, un limite inferiore e uno superiore. L'algoritmo classifica i pixel in base alla posizione tra il gradiente di un pixel stesso e queste due soglie.

a. Se il gradiente del pixel è maggiore al limite della soglia superiore, allora il pixel verrà accettato come appartenente ad un bordo

b. Se, invece, il gradiente è inferiore al limite inferiore, allora il pixel verrà scartato

c. Se, infine, il gradiente del pixel si trova tra le due soglie, allora il pixel verrà accettato solo se è collegato ad un pixel che appartiene ad un bordo

Purtroppo però anche questo metodo, che sulla carta risulta infallibile e lo stato dell'arte per l'individuazione dei bordi, non ha fornito i risultati sperati. Purtroppo le immagini continuano ad essere troppo rumorose e *Canny* individua come contorni un elevato numero di falsi positivi. Un risultato ottenuto con questo algoritmo si può osservare in Figura 2.7.

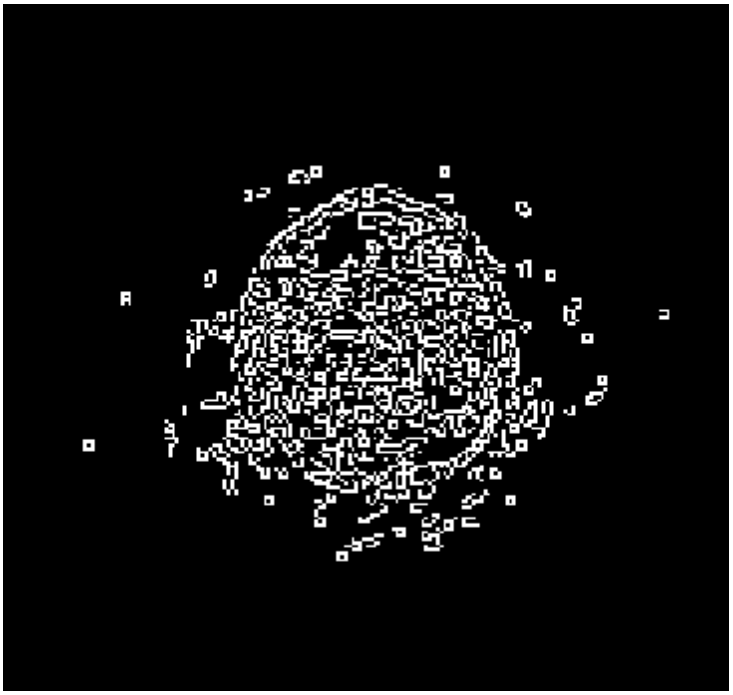


Figura 2.7: Risultato dell'applicazione del filtro di *Canny*

Capitolo 3

Soluzione del problema

Nelle sezioni a seguire verrà illustrato, passo dopo passo, il procedimento che ha portato alla soluzione del problema in esame. Verranno affrontati i diversi passaggi compiuti e, per ognuno di essi, verranno spiegate le scelte prese. Si partirà con lo spiegare le tecniche adottate nella fase di pre-processing: in questo frangente, infatti, le immagini vengono suddivise e scelte, in base al loro contenuto informativo, in maniera automatica. Successivamente verranno esposti i passaggi compiuti per giungere alla corretta segmentazione del contorno, utilizzato poi per il riconoscimento del movimento. Nella sezione successiva si parlerà di come avviene, effettivamente, il riconoscimento dell'angolo di inclinazione della sagoma segmentata. Infine si potrà capire come avviene il processo decisionale che andrà ad attestare l'effettivo movimento del paziente.

3.1 Pre-processing

Come spiegato nel capitolo precedente, l'acquisizione delle immagini non avviene in modalità *frame* ma, bensì, in *list-mode*. Ciò comporta, oltre tutti gli altri aspetti già trattati, la presenza di un elevato numero di immagini per ogni acquisizione. Tali immagini, fornite come output dalla macchina, vengono salvate in un'unica cartella senza alcun tipo di distinzione, se non il nome. Ai fini della successiva analisi delle stesse, si è reso necessario un passaggio intermedio atto ad una più ponderata organizzazione dei file. A tale scopo è stato creato uno script in Python che, partendo dal percorso contenente tutti i file DICOM da analizzare, li suddivide nei rispettivi frame e ne

effettua il *reslicing* lungo i tre assi: sagittale, coronale e assiale. Per entrare nel merito del funzionamento della procedura di reslicing bisognerà attendere il paragrafo successivo. Python è un linguaggio di scripting *object oriented* ed è, quindi, una combinazione di flessibilità, data dal fatto che non deve essere compilato, e potenza, tipica dei linguaggi orientati agli oggetti. Python, inoltre, dispone di una larghissima community di sviluppatori che, negli anni, ne hanno aiutato la crescita, la stabilità e a renderlo un linguaggio sempre più completo arricchendolo di numerose estensioni. [20] Esso presenta, infatti, numerose classi e funzioni per assolvere *task* apparentemente semplici ma che, invece, richiederebbero un enorme sforzo se implementati usando linguaggi di programmazione più ad alto livello quali, ad esempio, C/C++. Per queste motivazioni, dunque, Python si è rivelato essere sicuramente il candidato ideale per la prima fase di organizzazione.

Ricapitolando, quindi, le operazioni eseguite dallo script Python sono sostanzialmente due: organizzazione e *reslicing*. Nella prima fase, quella di organizzazione, lo script passa in rassegna tutti i file DICOM contenuti nella cartella che gli viene passata come argomento in ingresso. Ogni file DICOM, come accennato nel capitolo introduttivo, porta con sé diverse informazioni: oltre ai dati *raw* dell'immagine, infatti, tali file presentano un *header* che contiene diverse informazioni quali, ad esempio, l'indice che quell'immagine ha all'interno della serie e quanti immagini sono contenute in ogni serie. Con queste due informazioni, e qualche banale calcolo matematico, è possibile suddividere i file DICOM nei rispettivi *frame*. La maggior parte degli studi utilizzati per validare e testare il corretto funzionamento del programma oggetto di questa tesi sono composti da 25 frame ognuno costituito da 127 file DICOM. Ognuno di questi file contiene informazioni su un'immagine, su una *slice*, di dimensione 344×344 pixel. Come è possibile iniziare ad intuire, e come è reso maggiormente chiaro dalla rappresentazione in Figura 3.1, tali slice rappresentano un volume di dimensione $344 \times 344 \times 127$. Da questo volume, dunque, è possibile esportare le tre diverse viste: assiale, coronale e sagittale. Tale procedimento, detto *reslicing*, costituisce la seconda fase dello script Python. Esso, quindi, a partire dal volume DICOM estrae, per ogni vista, le slice corrispondenti e le salva in tre cartelle distinte (x, y e z) in formato Joint Photographic Experts Group (JPEG). Questa operazione, inoltre, viene eseguita per ogni frame. In questo modo, alla fine del processo di reslicing, nella cartella x e y del frame i-esimo, saranno presenti 344 immagini di dimensione 344×127 pixel che rappresentano, rispettivamente, la vista coronale e quella sagittale mentre nella cartella z vi saranno 127 immagini di dimensione 344×344 pixel rappresentanti la vista assiale.

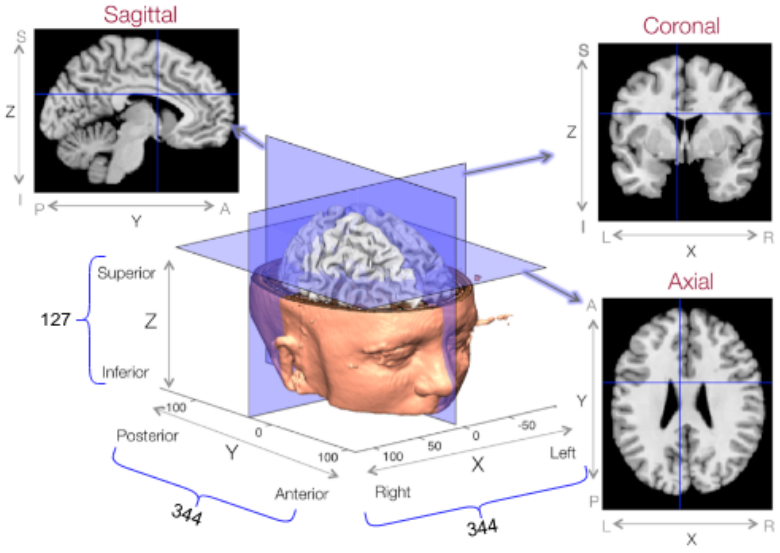


Figura 3.1: Rappresentazione grafica della suddivisione in slice di un volume DICOM

Uno dei requisiti fondamentali che è stato richiesto di rispettare in fase di progettazione riguarda il livello di automazione del programma, l'interazione con l'operatore deve essere minima o, ancora meglio, assente. Si giunge così al primo problema decisionale che il software si trova a dover affrontare: bisogna scegliere, tra le immagini a disposizione per ogni vista, quali utilizzare al fine del riconoscimento del movimento. Il problema nasce dal fatto che, quando si esegue un esame diagnostico tramite MR, l'inizio della scansione effettuata non comincia necessariamente con l'inizio della parte da analizzare, la testa in questo caso, né è possibile determinare o stimare a priori quanto spazio intercorra tra le due entità. Ciò si traduce, a livello di immagini, con una serie di slice che non contengono alcun tipo di informazione riguardante il soggetto in esame. Nel mondo delle immagini, l'informazione che un'immagine trasporta è valutabile tramite l'intensità dei pixel dell'immagine stessa. [21] L'entropia di un'immagine infatti, ovvero il grado di informazione in essa contenuto, può essere calcolata, secondo il teorema dell'entropia di Shannon

$$H(X) = \sum_i P(x_i) I(x_i) \tag{3.1}$$

Dato che le immagini in analisi sono rappresentate da un solo cana-

le e, quindi, in scala di grigi, il contenuto informativo può essere confuso con la somma dell'intensità di ogni singolo pixel divisa poi per il numero di pixel contenuti nell'immagine: ovvero con la media. Ciò è verificato se si considerano i pixel neri, ovvero quelli di intensità 0, come pixel a contenuto informativo nullo. Questo rispecchia proprio il nostro caso in quanto tali pixel rappresentano lo sfondo dell'immagine. In Figura 3.2 e 3.3 è riportata, per ogni slice e per ogni frame, l'intensità media delle immagini di un paziente. Vengono prese in considerazione solamente la vista coronale e quella assiale in quanto la vista sagittale è paragonabile a quella coronale. Per permettere al programma di decidere in modo autonomo da quale immagine cominciare l'analisi sono state effettuate diverse prove e misurazioni e si è giunti, infine, alla definizione di un parametro di progetto che identifica la soglia minima che deve presentare un'immagine per essere analizzata. Tale soglia è stata impostata a 3.0. Il programma, quindi, scarta tutte le immagini che hanno soglia minore di 3.0, mentre analizza le N immagini successive alla prima che presenta un valore maggiore di quella soglia. Il numero N di immagini varia caso per caso, l'algoritmo riterrà conclusa la fase di analisi non appena incontrerà 3 immagini che presentano un'intensità media minore di quella della soglia, ovvero quando si troverà di fronte ad un fronte di discesa.

3.2 Segmentazione

Figura 3.4: Rappresentazione di un'immagine e della relativa ampiezza dello spettro in base logaritmica

Le immagini che vengono selezionate al passo precedente sono pronte per essere sottoposte alla procedura di segmentazione. Durante questa fase viene eliminato il rumore presente intorno al contorno, l'immagine viene trasformata da scala di grigi con 8 bit di profondità in un'immagine binaria e, infine, il contorno viene estrapolato. Procedendo con ordine, la prima fase è quella di eliminazione, o se non altro attenuazione, del rumore. Nel capitolo precedente sono state esposte alcune strade, infruttuose, tentate per giungere a tale scopo. Purtroppo nessuna di esse aveva fornito dei risultati soddisfacenti. L'unica strada che è risultata efficace per lo scopo prefissato è stata l'applicazione di un Low Pass Filter (LPF). Per fare ciò, per applicare un filtro passa basso, bisogna, quasi inevitabilmente, passare dal dominio dello spazio a quello della fase. Tale trasformazione avviene mediante l'utilizzo della trasformata di Fourier. Come si può osservare in Figura 3.4, la trasformazione di un'immagine nel dominio della frequenza produce, a sua volta, un'immagine che contiene componenti a tutte le frequenze. Le ampiezze, però, diminuiscono all'aumentare delle frequenze. Perciò, le componenti a bassa frequenza contengono più informazioni, relative all'immagine, rispetto a quelle ad alta frequenza. Dall'immagine trasformata, inoltre, è possibile osservare come esistano due direzioni dominanti nella trasformata di Fourier: una verticale e una orizzontale, entrambe passanti per il centro. Ciò è dovuto, sostanzialmente, al fatto che lo sfondo dell'immagine risulta abbastanza regolare. Siccome vengono prese in considerazione solo immagini digitali, la trasformata di Fourier che verrà utilizzata è quella discreta (Discrete Fourier Transform (DFT)). Per un'immagine di dimensione $N \times N$ la DFT risulta essere:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi(\frac{ki}{N} + \frac{li}{N})} \tag{3.2}$$

dove $f(a,b)$ rappresenta l'immagine nel dominio della frequenza mentre il termine esponenziale è la funzione base corrispondente ad ogni punto $F(k,l)$ dello spazio di Fourier. L'equazione può quindi essere interpretata come: il valore di ogni punto $F(k,l)$ è ottenuto moltiplicando l'immagine spaziale con la corrispondente funzione base ed eseguendo poi la somma dei risultati. Le funzioni base sono co-sinusoidi di frequenza crescente, ad esempio $F(0,0)$ rappresenta la componente continua dell'immagine che corrisponde all'intensità media mentre $F(N-1, N-1)$ rappresenta la frequenza maggiore. [22]

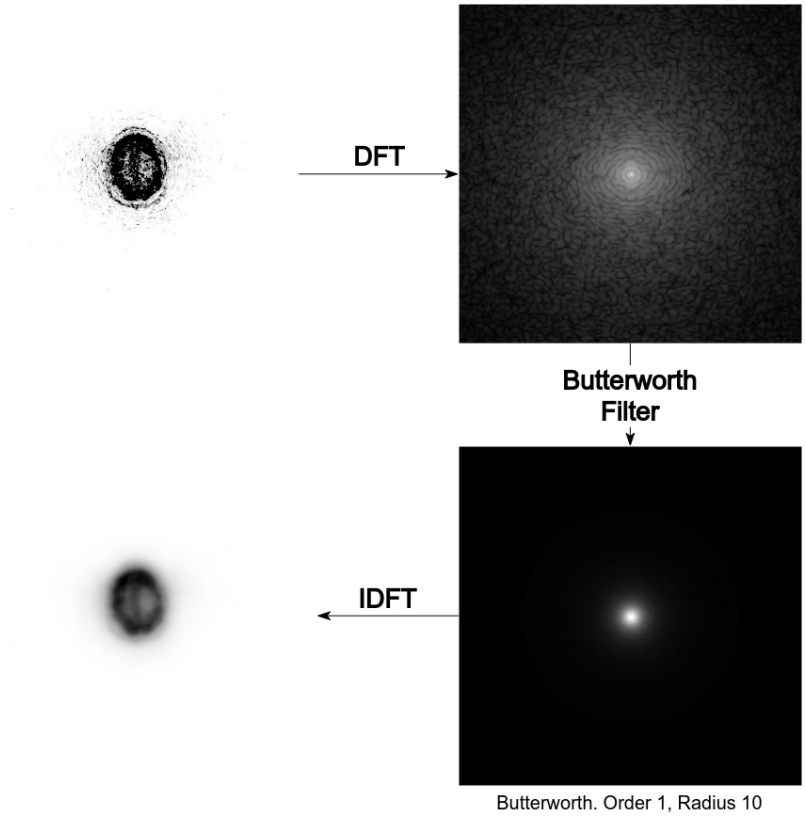


Figura 3.5: Flusso logico di lavoro del filtro di Buttleworth

Una volta calcolata la DFT dell'immagine si può procedere con l'applicazione di un filtro passa basso. A tale scopo si è deciso di utilizzare il filtro di Butterworth in quanto migliore approssimazione di un LPF ideale. Lo scopo di tale filtro è ottenere una risposta in frequenza il più possibile piatta, in modulo, nella banda passante. Il nome del filtro deriva dall'autore dell'articolo *On the Theory of Filter Amplifiers*, Stephen Butterworth, primo a descriverlo pubblicamente. [23] La formulazione matematica del filtro di Buttleworth è descritta nell'equazione 3.3.

$$H(f) = \frac{B(f)}{A(f)} = \frac{b_1(jf)^n + b_2(jf)^{n-1} + \cdots + b_{n+1}}{a_1(jf)^n + a_2(jf)^{n-1} + \cdots + a_{n+1}} \tag{3.3}$$

L'implementazione di tale filtro accetta in ingresso due parametri: il raggio D e l'ordine n. Anche questi due parametri, come prima di loro la soglia, sono stati decisi a seguito di varie prove e i loro valori si attestano su

10, per quanto riguarda il raggio, e 1 per l'ordine. Un esempio di applicazione del filtro è illustrato in Figura 3.5.

A questo punto, dopo l'applicazione del filtro, comincia la fase di segmentazione del bordo. Sempre tenendo in considerazione il giusto compromesso tra efficienza e precisione, si è deciso di utilizzare il metodo `findContours` della libreria OpenCV. Tale metodo sfrutta l'algoritmo descritto da Suzuki e Abe nell'articolo *Topological Structural Analysis of Digitized Binary Images by Border Following.* [24] Nell'articolo, gli autori, espongono due nuovi algoritmi in grado di individuare e ì seguire i bordi all'interno un'immagine binaria. Molti articoli avevano già trattato l'argomento dell'analisi topologica e strutturale di immagini binarie, le tecniche fino ad allora utilizzate, però, prevedevano la scansione orizzontale (ossia *raster*) dell'immagine seguita da una procedura di etichettatura, o *labeling.* Nessuno ancora però aveva pensato di utilizzare la tecnica del *border following* per condurre una tale analisi. Gli autori si sono quindi ispirati all'algoritmo per l'individuazione del bordo proposto da Rosenfeld e Kak [25] migliorandolo ed introducendo le seguenti estensioni

1. ogni bordo individuato presenta un identificativo univoco, mentre nell'algoritmo originale tutti i bordi presentavano lo stesso identificativo

2. è stata introdotta una funzione in grado di ottenere, a partire dal bordo in esame, il suo genitore

L'algoritmo, come accennato, basa il suo funzionamento sulle immagini binarie. Tali immagini presentano pixel la cui densità può essere solamente o 0 o 1, essi sono chiamati rispettivamente 0-pixel e 1-pixel. Senza alcuna perdita di generalità, inoltre, si assume che gli 0-pixel costituiscano lo sfondo dell'immagine binaria. Una componente connessa di 1-pixel e di 0-pixel viene chiamata, rispettivamente, 1-component e 0-component. Se una 0-component S contiene il bordo dell'immagine, tale componente S verrà identificata come lo sfondo; altrimenti sarà considerata come un *buco.* Per applicare il metodo `findContours` bisognerà, quindi, trasformare l'immagine di partenza in un'immagine binaria. Per fare ciò, ancora una volta ci viene in aiuto OpenCV che, tra i vari metodi per applicare una soglia, presenta anche quello binario. Applicare una soglia di tipo binario significa, quindi, cambiare in 1 l'intensità di tutti i pixel il cui valore è maggiore della soglia e in 0 gli altri. In questo modo si otterrà un'immagine binaria a cui si potrà applicare l'algoritmo descritto in [24].

A partire dall'output del filtro di Buttleworth, quindi, si applica una soglia il cui valore è stato deciso a seguito delle prove effettuate sul dataset. Tale valore è un ulteriore parametro di progetto ed è stato posto uguale a 236, che rappresenta il 92.55% dell'intensità massima ovvero di 255. Il valore della soglia è stato mantenuto relativamente elevato perché, grazie all'azione del filtro applicato in precedenza, la componente informativa dell'immagine a cui si intende applicare la soglia corrisponde ad intensità elevate mentre la componente rumorosa è confinata ad intensità più basse. L'equazione della soglia applicata all'immagine è riportata in 3.4

$$\mathrm{dst}(x,y) = \begin{cases} 1 & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Alcuni risultati dell'applicazione della soglia appena descritta, in cascata al filtro di Buttleworth, sono visionabili in Figura 3.6. Dalle immagini si può notare come il rumore sia completamente sparito mentre il contorno risulti, ora, nettamente distinguibile e individuabile.

(a) Vista coronale **(b)** Vista sagittale

Figura 3.6: Esempio di applicazione di una soglia binaria

Dato che le immagini ora sono state rese binarie, è possibile eseguire l'algoritmo di Suzuki e Abe per individuarne il contorno. L'algoritmo è molto efficiente e preciso, applicandolo alle due immagini di Figura 3.6 si ottengono i risultati riportati in Figura 3.7. Come si può notare, il contorno viene riconosciuto alla perfezione e segmentato con estrema precisione. Nel caso, nonostante tutte le operazioni di filtraggio, fosse ancora presente del rumore questo verrebbe riconosciuto come un contorno esterno. Per questo, l'algoritmo sviluppato in questa tesi, passa in rassegna tutti i contorni trovati e, sulla base di alcune valutazioni come la dimensione e la posizione, decide quali di questi rappresenti effettivamente il bordo desiderato.

(a) Vista coronale **(b)** Vista sagittale

Figura 3.7: Esempio di applicazione dell'algoritmo di Suzuki e Abe alle immagini di Figura 3.6

3.3 Riconoscimento dell'angolo

Una volta segmentato correttamente il bordo del cranio nelle tre diverse viste giunge il momento di rilevare se vi sia stato, o meno, un movimento. A tale scopo bisogna identificare e determinare un angolo o uno spostamento a partire dal contorno. L'idea di base è, quindi, calcolare per ogni *slice* di ogni *frame* l'angolo di inclinazione del bordo e, successivamente, calcolare l'angolo medio di ogni *frame* e confrontare quest'ultimo tra *frame* successivi per determinare l'effettivo movimento. Quindi, ricapitolando, per ogni *slice* appartenente all'i-esimo *frame* verrà calcolato un angolo e questo concorrerà a determinare l'angolo medio del *frame* i. Questo procedimento viene applicato per ogni *frame* all'interno del dataset e per ognuna delle tre viste. Perciò, alla fine del processo, ogni *frame* sarà caratterizzato da 3 angoli ognuno dei quali relativo ad una delle 3 viste: coronale, sagittale e assiale. Dal confronto di questi angoli verrà, successivamente, dedotto e ricostruito il possibile movimento del paziente in esame. Per calcolare l'angolo di rotazione del contorno appena individuato si è ricorso all'utilizzo dei momenti.

I momenti vengono utilizzati per descrivere globalmente il *layout* di una forma, essi combinano, infatti, diversi descrittori come l'area, la compattezza, l'irregolarità. Sono molto efficaci nella descrizione in quanto risultano ampiamente tolleranti al rumore. Nel mondo dell'analisi delle immagini i momenti che vengono utilizzati sono quelli di tipo statistico e non meccanico anche se, tra i due tipi di momenti, ci sono grandi analogie. Per esempio, il momento meccanico di inerzia descrive l'inerzia di un corpo al variare della sua velocità rotazionale; mentre il momento statistico del secondo ordine descrive l'area di un contorno al variare dello stesso. Quindi, i momenti statistici possono essere considerati come descrittori globali di aree. Essi sono stati introdotti, nell'analisi delle immagini, negli anni '60 da Hu. [26] I momenti, in realtà, sono maggiormente associati al riconoscimento di *pattern*

di tipo statistico rispetto a quelli basati su modelli visivi perché una delle maggiori assunzioni è che, per lavorare appieno, essi abbiano bisogno di una visione non occlusa della forma da analizzare. Il punto di forza dei momenti è sicuramente il fatto che essi forniscano una descrizione dell'oggetto che non dipende, ed è quindi invariante, rispetto alla rotazione e alle dimensioni dello stesso. [27]

Il momento di una funzione continua bidimensionale, di ordine $p+q$, può essere espresso come

$$m_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \tag{3.5}$$

Il teorema di unicità afferma che se $f(x,y)$ è continua a tratti ed ha valori non nulli solo in una porzione finita del piano xy, allora esistono i momenti di ogni ordine e la sequenza dei momenti $(m_{p,q})$ è unicamente determinata da $f(x,y)$. Vale anche il viceversa, ossia: $m_{p,q}$ determina unicamente $f(x,y)$. In pratica, la funzione può essere descritta in funzione dei suoi momenti di ordine più basso. Nel caso di immagini, e quindi per funzioni di tipo discreto, la funzione $f(x,y)$ coincide con l'intensità che caratterizza i singoli pixel e, quindi, con la funzione $I(x,y)$. Perciò l'equazione 3.5 diventa

$$m_{i,j} = \sum_x \sum_y x^i y^j I(x,y) \tag{3.6}$$

I momenti rappresentano, quindi, una particolare media dell'intensità dei pixel che compongono l'immagine. Il momento di ordine zero, $m_{0,0}$, è

$$m_{0,0} = \sum_x \sum_y I(x,y) \tag{3.7}$$

che rappresenta l'area, per immagini binarie, o la somma dei livelli di grigio per immagini in scala di grigio. Mentre i due momenti del primo ordine, $m_{0,1}$ e $m_{1,0}$, vengono calcolati tramite

$$m_{1,0} = \sum_x \sum_y xI(x,y) \qquad m_{0,1} = \sum_x \sum_y yI(x,y) \tag{3.8}$$

Nel caso di immagini binarie, questi valori sono proporzionali alle coordinate del centro della forma considerata (per ottenere le coordinate esatte basta dividere tali valori per l'area della forma). In generale, il centro di massa $(\bar{x}, \bar{y})$ può essere calcolato come il rapporto tra la componente del primo ordine e quella dell'ordine zero:

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}} \qquad \bar{y} = \frac{m_{0,1}}{m_{0,0}} \tag{3.9}$$

L'equazione 3.5 rappresenta i cosiddetti momenti spaziali, un'altra forma di momenti spesso utilizzata è quella centrale. I momenti centrali possono essere derivati da 3.5 semplicemente dividendo i momenti spaziali con il centro di massa $(\bar{x}, \bar{y})$ 3.9 dell'oggetto. In questo modo tutti i momenti saranno riferiti al centro di gravità dell'oggetto. Tali momenti possono essere calcolati mediante la seguente formula

$$\mu_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy \tag{3.10}$$

Da 3.10 si possono facilmente derivare le due seguenti relazioni

$$\begin{aligned} \mu_{0,0} &= m_{0,0} \\ \mu_{1,0} = \mu_{0,1} &= 0 \end{aligned} \tag{3.11}$$

Inoltre i momenti centrali possono sempre essere espressi in funzione di quelli spaziali:

$$\mu_{p,q} = \frac{m_{p,q}}{m_{0,0}} - \left(\frac{m_{1,0}}{m_{0,0}}\right)^p * \left(\frac{m_{0,1}}{m_{0,0}}\right)^q \tag{3.12}$$

I momenti, dunque, possono essere considerati alla stregua di vere e proprie caratteristiche dell'oggetto in analisi, che ne permettono una ricostruzione statistica e geometrica. I momenti di per sé non presentano un significato geometrico ma, da essi, possono essere derivate alcune proprietà geometriche dell'oggetto. Maggiore è la complessità dell'oggetto analizzato maggiore dovrà essere l'ordine dei momenti al fine di riuscire a descrivere l'oggetto ed essere, successivamente, in grado di ricostruirlo tramite essi. Per

gli scopi di questa tesi verranno utilizzati solo momenti di ordine massimo due. Sono già state esposte alcune proprietà geometriche che possono essere derivate dai momenti: l'area 3.7 e il centro di massa 3.9. Per quanto riguarda l'orientazione, invece, essa è definita come l'inclinazione tra l'asse x e l'asse attorno al quale l'oggetto può essere ruotato applicando il momento di inerzia minore, solitamente quest'asse coincide con il semiasse maggiore. In questa direzione l'oggetto presenta la sua massima estensione. L'orientazione, θ, può essere espressa come

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} \tag{3.13}$$

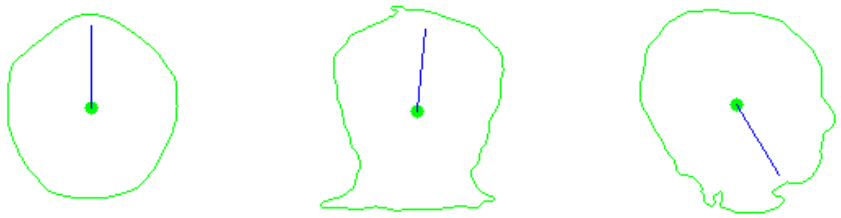
L'equazione 3.13 porta ad una sorta di ambiguità nel calcolo dell'orientazione θ. Questa ambiguità può essere risolta scegliendo sempre θ come l'angolo tra l'asse x e il semiasse maggiore a, tale per cui $a \geq b$. [28]

Di seguito viene riportato il codice C++ utilizzato per il calcolo dell'orientazione

```
Moments m = mu[contour_idx];

double theta = 0;
double mu20_mu02 = (m.mu20 - m.mu02);
if ((mu20_mu02 == 0) & (m.mu11 == 0))
  theta = 0;
else if ((mu20_mu02 == 0) & (m.mu11 > 0))
  theta = M_PI / 4;
else if ((mu20_mu02 == 0) & (m.mu11 < 0))
  theta = -M_PI / 4;
else if ((mu20_mu02 > 0) & (m.mu11 == 0))
  theta = 0;
else if ((mu20_mu02 < 0) & (m.mu11 == 0))
  theta = -M_PI / 2;
else if ((mu20_mu02 > 0) & (m.mu11 > 0))
  theta = 0.5 * atan((2 * m.mu11) / mu20_mu02);
else if ((mu20_mu02 > 0) & (m.mu11 < 0))
  theta = 0.5 * atan((2 * m.mu11) / mu20_mu02);
else if ((mu20_mu02 < 0) & (m.mu11 > 0))
  theta = 0.5 * atan((2 * m.mu11) / mu20_mu02) + M_PI / 2;
else if ((mu20_mu02 < 0) & (m.mu11 < 0))
  theta = 0.5 * atan((2 * m.mu11) / mu20_mu02) - M_PI / 2;
```

3.4 Analisi dell'orientazione



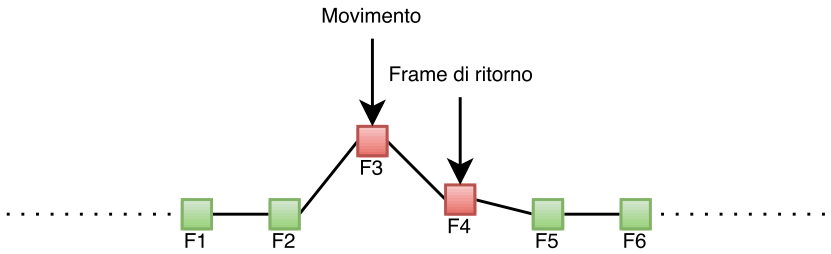
(a) Vista assiale **(b)** Vista coronale **(c)** Vista sagittale

Figura 3.8: Individuazione dell'angolo di rotazione nelle tre viste

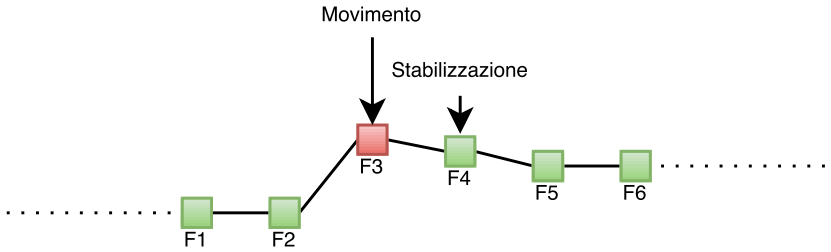
Una volta calcolata l'orientazione di ogni bordo di ogni *slice* di ogni *frame* per ogni vista bisognerà decidere se e quando si è in presenza di movimento e, successivamente, quantificarlo. Per procedere con questa delicata operazione si è deciso di sfruttare tutte e tre le viste per decidere al fine di individuare precisamente il movimento del paziente. Per ogni vista, dunque, verranno marcati i *frame* in cui c'è stato, possibilmente, un movimento del soggetto. Se tale movimento verrà rilevato anche nello stesso *frame*, in una delle altre due viste, allora il programma, oggetto di questa tesi, evidenzierà all'operatore che a tale istante c'è stato un movimento. L'operatore, a quel punto, avrà il compito di decidere che come comportarsi: potrà scartare tutti i dati successivi, o precedenti, a quell'istante oppure far rifare l'esame al paziente. Il punto di forza di questo processo è la velocità di esecuzione del programma. Per analizzare uno studio completo, composto da oltre tremila *slice*, il programma impiega meno di 5 minuti. Questo aspetto verrà ripreso in maniera più approfondita nei capitoli successivi.

Come è riportato in Figura 3.8 l'orientazione delle tre viste non è, come ci si potrebbe aspettare, coerente. Essa però risulta coerente all'interno di ogni singola vista, perciò esse devono essere analizzate separatamente e, solo successivamente, i risultati possono essere confrontati. La spiegazione della non coerenza dell'orientazione risiede nel metodo utilizzato per calcolare la stessa. Come spiegato in precedenza, infatti, l'angolo viene calcolato attraverso i momenti ed ha un forte legame con l'area della figura in esame. Essendo le tre viste tre rappresentazioni diverse, seppure dello stesso volume, è facile intuire come esse presentino area diversa quindi una diversa distribuzione ed un diversa orientazione.

Il procedimento di analisi di ogni singola vista basa il suo funzionamento sul calcolo di una media mobile. La finestra utilizzata per selezionare i campioni di cui calcolare la media non ha lunghezza prefissata bensì risulta essere di tipo adattivo. Per identificare i *frame* in cui, ragionevolmente, il soggetto si è mosso si parte utilizzando una finestra di 3 elementi. Di questi tre elementi si calcola media, μ, e deviazione standard, std, se tutti e tre i valori ricadono all'interno dell'intervallo $[\mu - 3 * std, \mu + 3 * std]$ allora si continua ad allargare la finestra e calcolare media e deviazione standard dei valori in essa presenti finché uno non ricade fuori da tale intervallo. Se tutti gli elementi, che rappresentano la media degli angoli di orientazione calcolati per ogni *frame*, ricadono all'interno del *range* $\mu \pm 3 * std$ ciò significherà che non vi è stato movimento significativo. Se, invece, durante l'analisi si riscontra un valore che giace all'esterno dell'intervallo considerato bisognerà decidere come procedere. Sicuramente il *frame*, il cui valore dell'angolo di orientazione non risulta in linea con i precedenti valori, verrà marcato come *frame* in cui è avvenuto un movimento. Ora bisogna però distinguere due casi: il paziente potrebbe essersi mosso ed essersi poi stabilizzato, assumendo magari una posizione più comoda, oppure potrebbe verificarsi un movimento di *ritorno* in cui il soggetto ritorna, a meno di un certo errore, in una posizione simile a quella di partenza. Questi due casi vengono riconosciuti dal programma analizzando il *frame* in cui è stato rilevato il movimento e due *frame* ad esso successivi. Se tali *frame* risultano congruenti allora significa che il paziente, dopo il movimento, si è stabilizzato. La statistica precedente viene azzerata e ricalcolata a partire dal primo *frame* successivo a quello in cui vi è stato il movimento. Nel secondo caso, invece, il *frame* successivo a quello mosso viene marcato come *frame* di ritorno, e quindi *frame* in cui il paziente si è mosso. La statistica viene azzerata e ricomincerà dal secondo *frame* dopo il primo in cui si era rilevato il movimento. Il concetto appena espresso è rappresentato graficamente in Figura 3.9.



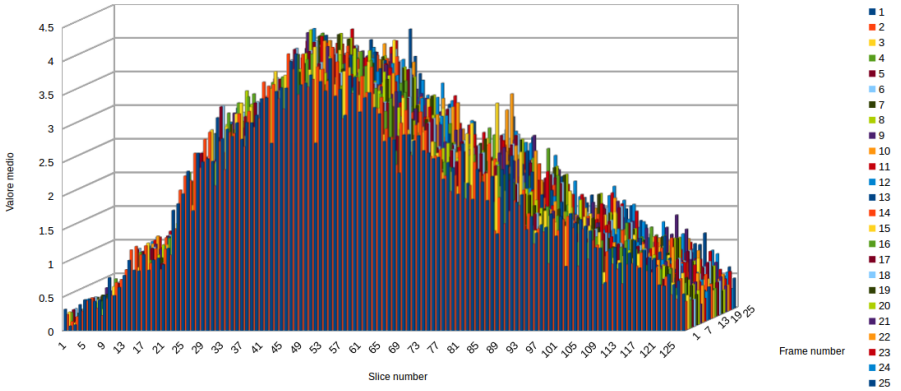
(a) Esempio di movimento con ritorno



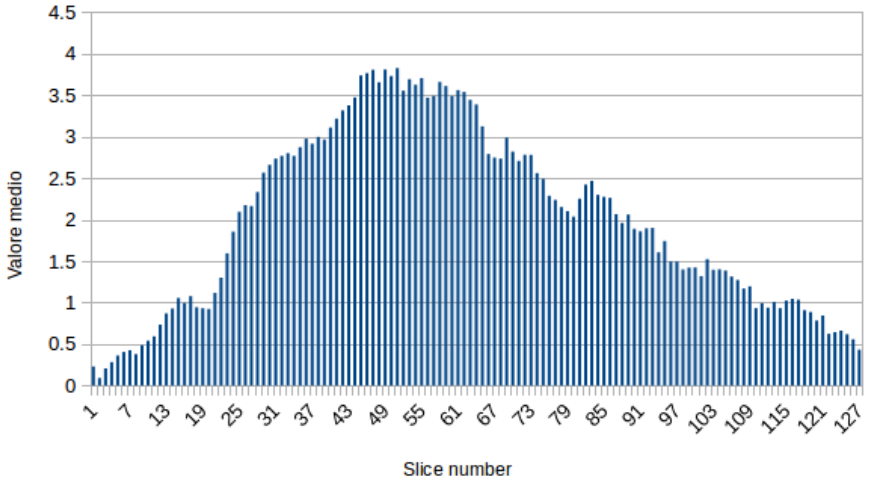
(b) Esempio di movimento con successiva stabilizzazione

Figura 3.9: Possibili configurazioni del movimento di un paziente

Con questo tipo di analisi, quindi, vengono evidenziati i *frame* in cui, presumibilmente, il paziente si è mosso. Una volta esposto tale risultato sarà premura dell'operatore decidere cosa fare. Se i *frame* che presentano movimento sono presenti in numero esiguo e localizzati all'inizio o alla fine dell'acquisizione, l'operatore potrebbe decidere di scartare tale sezione e considerare solo i *frame* esenti dal movimento. Oppure potrebbe far ripetere immediatamente l'esame al soggetto, nella sua totalità o solamente in parte.

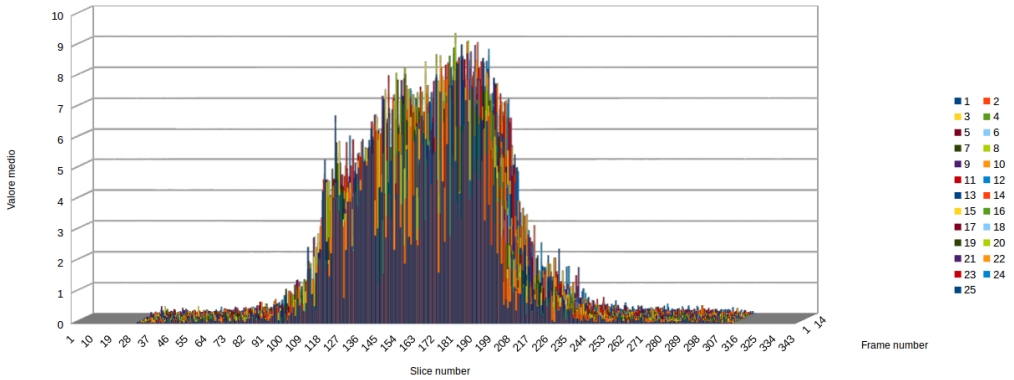


(a) Valore intensità medio assiale

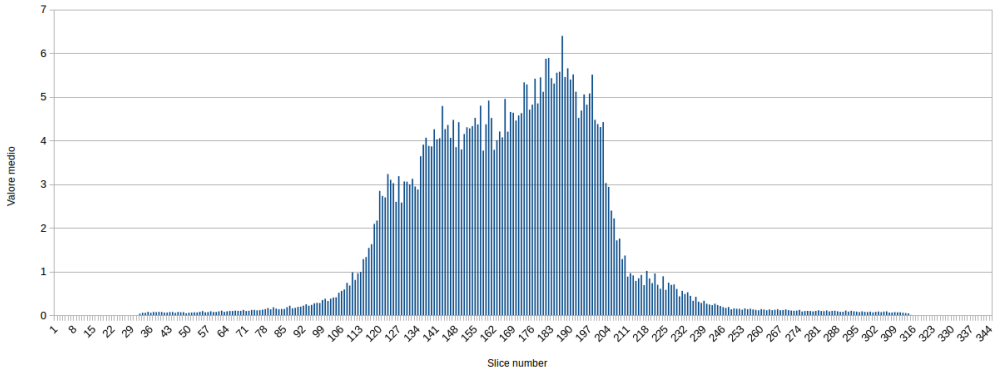


(b) Inviluppo medio assiale

Figura 3.2: Rappresentazione dell'intensità media delle immagini lungo la vista assiale, per ogni frame, per ogni slice



(a) Valore intensità medio coronale



(b) Inviluppo medio coronale

Figura 3.3: Rappresentazione dell'intensità media delle immagini lungo la vista coronale, per ogni frame, per ogni slice

Capitolo 4

Risultati

In questo capitolo verranno esposti i risultati ottenuti applicando il metodo descritto nel capitolo precedente. Prima, però, verrà analizzata la struttura del programma creato e i parametri da esso accettati. L'applicativo è stato progettato per essere di immediato e di semplice utilizzo, non richiede l'intervento dell'operatore né particolari conoscenze. Successivamente verranno visualizzati e analizzati i risultati prodotti dal software sui dataset a disposizione. Uno dei due insiemi di dati è stato generato a partire da quello contenente dati reali attraverso trasformazioni e rotazioni atte alla verifica del corretto funzionamento del programma. Questo processo si è reso necessario non avendo informazioni in grado di fornire un riscontro oggettivo sui movimenti effettuati dai pazienti del dataset reale.

4.1 Utilizzo del programma

Come precedentemente anticipato nella sezione Pre-processing, il programma è costituito da uno script Python e da un eseguibile `C++`. Lo script, finita la parte di pre-processing e trasformazione delle immagini dal formato medico DICOM a quello JPEG, avvia automaticamente l'eseguibile in cui è stato implementato il cuore dell'algoritmo di individuazione del movimento. Per avviare lo script Python bisognerà prima assicurarsi che abbia i corretti permessi di eseguibilità, tali permessi si concedono mediante il comando `chmod +x organize_dicom.py`. Successivamente lo si potrà avviare direttamente tramite il comando

```
$ ./organize_dicom.py
```

In Figura 4.1 è possibile vedere il messaggio di aiuto stampato sul terminale dal programma Python. Esso accetta in ingresso un percorso, il `path`, della cartella in cui sono presenti i file DICOM dell'acquisizione da analizzare. Se viene specificato il parametro `--no-move` questi file non verranno rinominati e spostati in sottocartelle, e quindi suddivisi nei rispettivi *frame*, ma tale organizzazione verrà riservata solo alle *slice* JPEG create. Questa opzione è molto utile se si vuole lasciare inalterata la cartella contenente i dati raw dell'acquisizione. Se, invece, in fase di avvio si specifica l'opzione `--debug` questa verrà passata all'eseguibile che mostrerà i vari passaggi intermedi, dall'applicazione del filtro alla segmentazione, a video. Per rendere maggiormente efficiente l'operazione di *reslicing* delle immagini dicom, essa viene eseguita in parallelo su più *thread*. Il numero dei quali viene deciso in base al numero dei *frame* da cui è composta l'acquisizione.

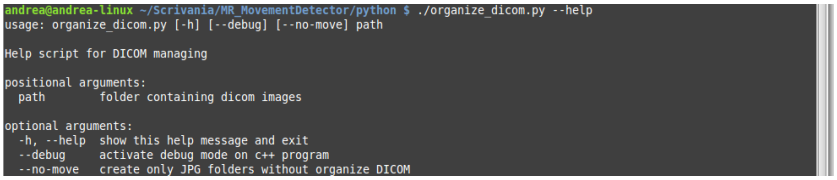


Figura 4.1: Schermata di *help* dello script Python

Mentre la parte Python non necessita di grandi preparativi per essere pronta ad essere eseguita, in quanto non deve essere compilata, il programma `C++` risulta più ostico nella fase iniziale. Prerequisito fondamentale per riuscire a compilare correttamente il codice sorgente del programma è quello di aver correttamente compilato e installato la libreria OpenCV. La spiegazione passo passo di come svolgere quest'operazione è stata trattata in precedenza nella sezione OpenCV. Una volta installata la libreria, quindi, si può procedere alla compilazione del programma tramite i seguenti comandi

```
$ mkdir build
$ cd build/
$ cmake ..
$ make
```

Terminata la fase di creazione del file binario, all'interno della cartella `build` si dovrebbe essere generato l'eseguibile `main`. Questo è l'eseguibile

che contiene tutta la logica per l'individuazione del movimento e che viene avviato dallo script Python.

Il codice sorgente è stato suddiviso in 3 file:

MovementDetector.h è l'*header* in cui vengono definite le funzioni implementate dalla classe

MovementDetector.cpp è il cuore del programma, è la classe che contiene tutta la logica del programma, dall'accesso alle immagini all'applicazione dei vari filtri fino alla segmentazione e output dei risultati

main.cpp è il file che contiene la funzione `main`, principale, e che istanzia la classe e la richima con i diversi parametri

Sfruttando la modularità del problema affrontato si è riusciti a creare funzioni comuni per la soluzione e l'ottenimento dei risultati dalle tre diverse viste. La classe `MovementDetector`, infatti, espone solo 3 metodi pubblici: il costruttore, `process_frame` e `get_results`. A seconda dei parametri passati al metodo `process_frame`, la classe saprà come aggiustare i diversi parametri a seconda che si stia analizzando una vista coronale, una sagittale o quella assiale. La firma di tale metodo è, infatti

```
int process_frame(const char *path, AxisType axis);
```

Il `path` è quello che è stato precedentemente passato allo script Python e che contiene le immagini in formato JPEG divise per *frame* e per vista. Per ogni *frame*, quindi, verrà processata la vista, o asse, specificata dal parametro `axis` di tipo `AxisType`. Successivamente potrà essere richiamato il metodo `get_results` che valorizzerà le variabili che gli sono state passate per riferimento con i risultati del processo. La firma di tale metodo è

```
1 void get_results( std::vector<double> &angle_variations );
```

Esso restituisce un vettore la cui lunghezza dipende dal numero di *frame* analizzati e ii cui valore nella posizione i-esima si riferisce all'orientazione media del *frame* i-esimo. Tale orientazione, quindi, viene calcolata per

ogni *slice* che compone il *frame* i-esimo. Di queste orientazioni viene successivamente calcolata la media: questo sarà il risultato presente nella posizione i-esima del vettore `angle_variations`.

Un esempio di utilizzo del codice è il seguente

```
bool detect_neck, rotate, debug;
vector<double> angle_variations;
char *path = "..";
MovementDetector mov = MovementDetector(detect_neck, rotate, debug);

mov.process_frame( path, MovementDetector::AXIAL );
mov.get_results(angle_variations);

mov.process_frame( path, MovementDetector::CORONAL );
mov.get_results(angle_variations);

mov.process_frame( path, MovementDetector::SAGITTAL );
mov.get_results(angle_variations);
```

4.2 Dataset fittizio (creato ad-hoc)

L'insieme dei dati utilizzati durante lo svolgimento del lavoro descritto in questo elaborato deriva da casi di studio reale. Ciò ha permesso, da una parte, di scontrarsi direttamente con tematiche reali che caratterizzano dati raccolti sul campo e non creati artificialmente. In questo modo, quindi, il programma realizzato può essere impiegato immediatamente, senza alcuna ulteriore modifica o calibrazione. Dall'altra parte, però, lavorare con dati reali significa non avere alcun tipo di controllo su di essi. Non si aveva alcun tipo di informazione riguardante il movimento presente all'interno di tali dati. A causa di questa mancanza, quindi, si è deciso di creare alcuni casi artificiali. A tale scopo, partendo da alcuni casi reali, si è introdotto del movimento in punti prestabiliti. In questo modo si è potuto valutare quantitativamente l'accuratezza e l'efficacia del processo di rilevamento del movimento.

Per validare il corretto funzionamento dell'algoritmo sono state effettuate diverse prove. Qui di seguito verranno riportate quelle più significative. Alcuni test prevedevano l'applicazione di un certo angolo ad un determinato *frame*. Altre prove invece sono state condotte simulando uno spostamento più reale che prevede una variazione mantenuta per un certo numero di *frame*. In tutte le prove eseguite il programma è stato in grado di individuare

correttamente il o i *frame* in cui era avvenuto lo spostamento e stimare, con un certo errore, tale variazione.

Nei grafici di Figura 4.2, 4.3 e 4.4 sono riportati i risultati prodotti dall'algoritmo sui dati del paziente 16. Sono stati introdotti 4 movimenti artificiali in corrispondenza dei *frame* 4, 6, 8 e 10 rispettivamente di 20, 10, 5 e 2 gradi. Come si può notare i movimenti artificiali vengono correttamente individuati mentre l'errore nella stima della variazione risulta essere di $\pm 2°$.

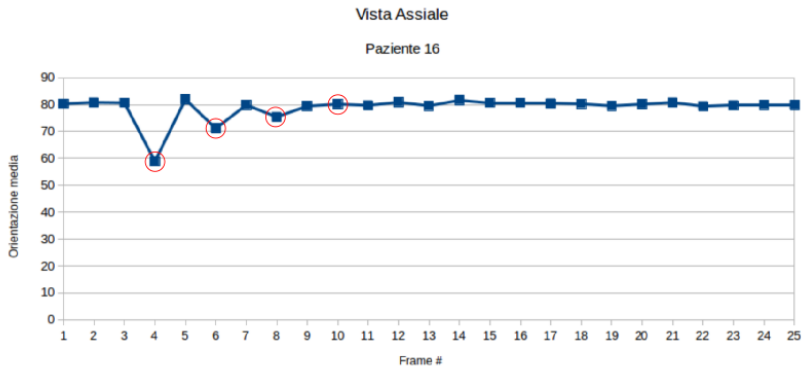


Figura 4.2: Vista assiale dei risultati prodotti dall'algoritmo sui dati del paziente 16. Sono stati introdotti 4 movimenti artificiali in corrispondenza dei *frame* 4, 6, 8 e 10 rispettivamente di 20, 10, 5 e 2 gradi

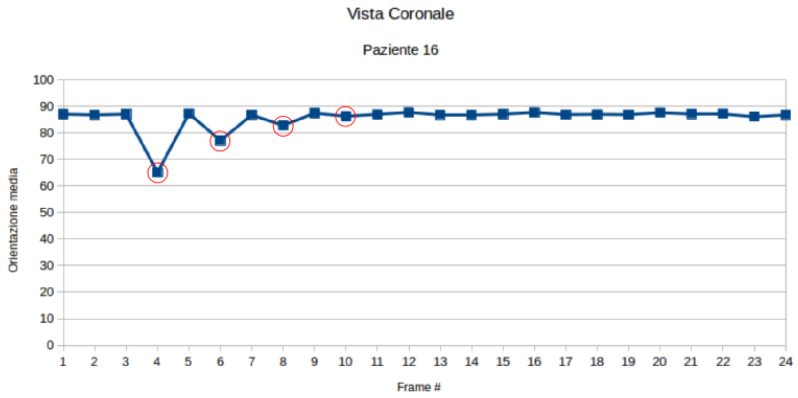


Figura 4.3: Vista coronale dei risultati prodotti dall'algoritmo sui dati del paziente 16. Sono stati introdotti 4 movimenti artificiali in corrispondenza dei *frame* 4, 6, 8 e 10 rispettivamente di 20, 10, 5 e 2 gradi

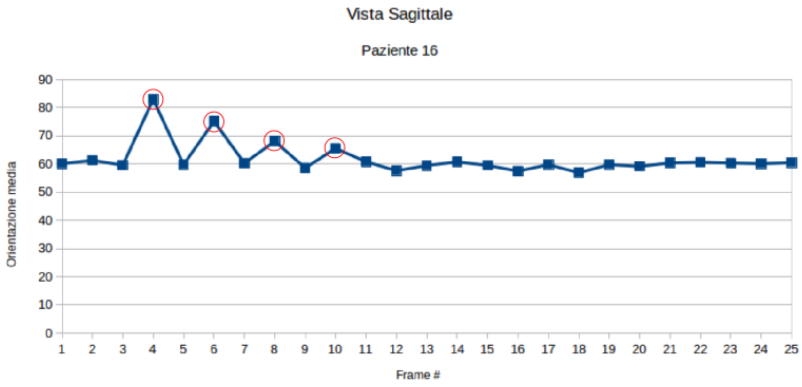


Figura 4.4: Vista sagittale dei risultati prodotti dall'algoritmo sui dati del paziente 16. Sono stati introdotti 4 movimenti artificiali in corrispondenza dei *frame* 4, 6, 8 e 10 rispettivamente di 20, 10, 5 e 2 gradi

Anche movimenti multipli vengono efficacemente riconosciuti. Al paziente 18, infatti, è stata applicato un movimento artificiale al *frame* 16 e uno al 21 rispettivamente di $+20$ e -10 gradi. Dal grafico in Figura 4.5 sono ben visibile i due movimenti, compreso quello di ritorno.

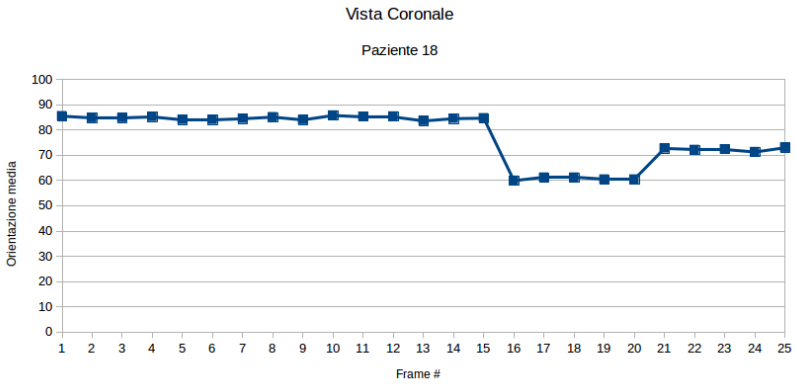


Figura 4.5: Vista coronale dei risultati prodotti dall'algoritmo sui dati del paziente 18 a cui sono stati applicati due movimenti ai *frame* 16 e 21. Il primo movimento è stato applicato in senso orario e il secondo in senso antiorario

I movimenti precedenti erano stati applicati singolarmente ad ogni vista e quindi non all'intero volume. Mentre è interessante notare come in Figura 4.6 i movimenti applicati a tutto il volume si ripercuotano anche sulle

altre viste. Nella prova, i cui risultati sono rappresentati in Figura 4.6, infatti, sono stati applicati molteplici movimenti

1. Frame 10: mosso di angolo significativo (in sagittale)
2. Frame 15: mosso di angolo significativo (in assiale)
3. Frame 20: mosso di angolo significativo (in coronale)
4. Frame 5: tutti i movimenti sopra descritti

Tutti questi movimenti vengono rilevati, in maniera più o meno evidente a seconda che la rotazione sia stata applicata alla vista in esame o ad una delle altre viste.

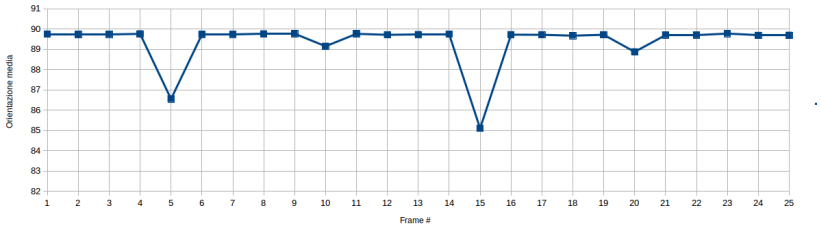


Figura 4.6: Vista assiale di un paziente a cui sono stati applicati movimenti multipli su tutte e tre le viste contemporaneamente

4.3 Dataset reale

I dati utilizzati all'interno di questo progetto provengono da pazienti che si sono sottoposti ad indagini diagnostiche mediante l'utilizzo di NMR. I dati sensibili di tali pazienti, quali nome, cognome, sesso ed età, sono stati rimossi dall'*header* dei file DICOM rendendoli, di fatto, anonimi. Queste non erano, comunque, informazioni che influiscono sul risultato finale dell'applicazione dell'algoritmo sviluppato durante questa tesi. Il dataset è composto da un totale di 10 pazienti anonimizzati e identificati mediante un numero: 16, 17, 18, 25, 27, 28, 29, 30, 32 e 33. Di questi soggetti, come già precedentemente accennato, non si ha una conoscenza quantitativa e qualitativa sull'effettivo movimento; non si sa, quindi, se questi pazienti si siano mossi durante il loro esame, di quanto e in che momento. I risultati prodotti dal programma,

inoltre, non hanno evidenziato movimenti significativi né se ne sono riusciti a riscontrare ad occhio nudo. In alcuni pazienti alcuni *frame* risulta che possano essere, ragionevolmente, *frame* in cui il paziente si è mosso. Tali movimenti però non sono valutabili ad occhio nudo né esiste uno strumento in grado di validare tale ipotesi. Nelle Figure 4.7, 4.8 e 4.9 sono riportati i grafici dei risultati prodotti dal programma sui 10 pazienti reali.

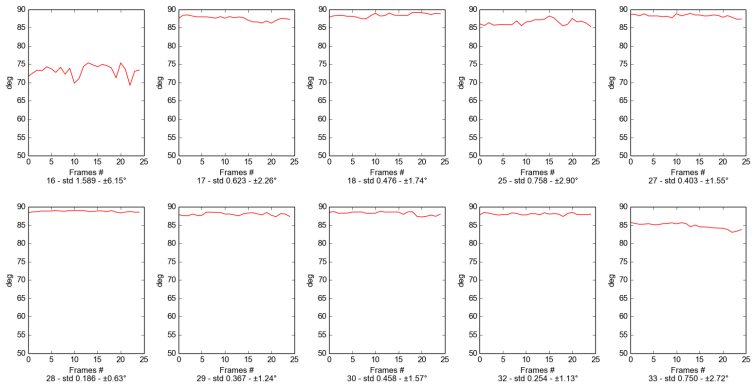


Figura 4.7: Risultati dell'algoritmo sui dati della vista assiale dei 10 pazienti reali

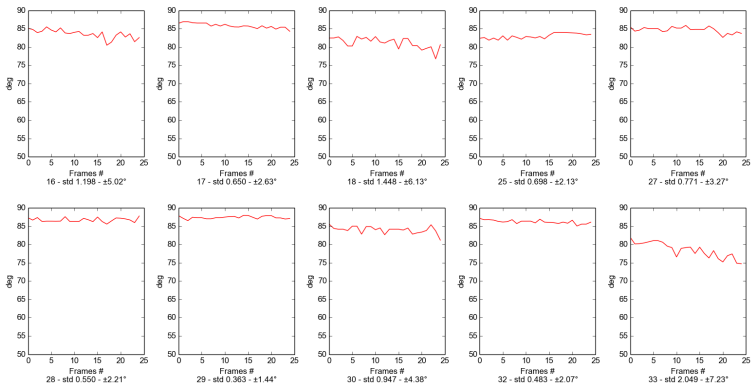


Figura 4.8: Risultati dell'algoritmo sui dati della vista assiale dei 10 pazienti reali

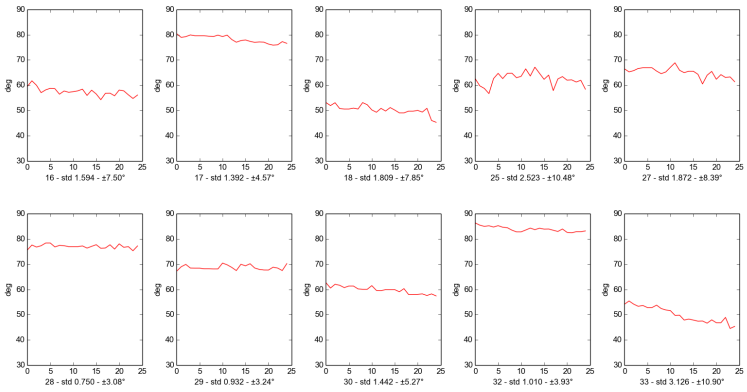


Figura 4.9: Risultati dell'algoritmo sui dati della vista assiale dei 10 pazienti reali

Purtroppo, non avendo un metro di paragone oggettivo e scientifico con il quale confrontare i risultati ottenuti sui pazienti reali, l'unica cosa che si può fare è avanzare delle considerazioni. La prima, e sicuramente più importante, è che i movimenti avvengono principalmente lungo l'asse sagittale. Ciò è confortante perché, quello sagittale, è il movimento che più facilmente riesce a compiere il paziente. Durante l'esame, infatti, i soggetti vengono bloccati mediante una bobina e dei cuscini che ne inibiscono, per quanto possibile, movimenti lungo l'asse assiale e quello coronale lasciandoli più liberi sul fronte sagittale. Un'altra osservazione è, quindi, che nei grafici relativi ai dati dei pazienti reali, Figure 4.7, 4.8 e 4.9, la vista assiale è quella che presenta meno spostamenti proprio a causa del modo in cui sono bloccati i soggetti durante lo svolgimento dell'esame.

4.4 Analisi della stima dell'errore

Conducendo ulteriori e più approfondite analisi sul funzionamento del programma è possibile riuscire a capirne i limiti. Come si è visto nei grafici riportati in Figura 4.3 e 4.4 la stima della variazione non è precisa. Nelle tabelle 4.2, 4.3 e 4.1 sono riportate le tabelle in cui vi è indicata la variazione applicata, quella riconosciuta dall'algoritmo e l'errore commesso. Inoltre, sempre dai grafici, si può notare come ci siano delle differenze nel grado di accuratezza dell'algoritmo da una vista all'altra. Mentre, nella vista coronale e in quella assiale, lo spostamento di 2° non viene rilevato, va a confondersi con la media, nella visione sagittale viene sovrastimato e, quindi, individua-

to. Ciò comporta la non individuazione di spostamenti inferiori ai 5mm nelle viste assiale e coronale, questo dato è stato ottenuto approssimando lo spostamento con il seno dell'angolo e moltiplicando il risultato per la lunghezza media di una testa umana che risulta essere tra i 20 e i 25 cm.

	Variazione applicata	Variazione misurata	Errore relativo
Frame 4	20°	21.42°	7.10%
Frame 6	10°	9.13°	8.70%
Frame 8	5°	4.88°	2.40%
Frame 10	2°	0.14°	93.00%

Tabella 4.1: Errore della stima della variazione lungo la vista assiale

	Variazione applicata	Variazione misurata	Errore relativo
Frame 4	20°	21.77°	8.85%
Frame 6	10°	9.85°	1.50%
Frame 8	5°	4.20°	16.00%
Frame 10	2°	0.77°	61.50%

Tabella 4.2: Errore della stima della variazione lungo la vista coronale

	Variazione applicata	Variazione misurata	Errore
Frame 4	20°	23.31°	16.55%
Frame 6	10°	15.54°	55.40%
Frame 8	5°	8.45°	69.00%
Frame 10	2°	5.91°	195.50%

Tabella 4.3: Errore della stima della variazione lungo la vista sagittale

Purtroppo, come emerge dai dati, l'errore nella stima della variazione dell'angolo è molto elevato. Ciò non permette una corretta ricostruzione del

movimento, che verrà infatti proposto come sviluppo futuro nel prossimo capitolo, ma solo un'individuazione dello stesso.

Capitolo 5

Conclusioni

Si è giunti, infine, al capitolo conclusivo. Nelle pagine seguenti verranno riassunti i risultati conseguiti, gli obbiettivi raggiunti, le possibili migliorie e gli sviluppi futuri. Particolare attenzione verrà posta su quali erano le richieste e come queste sono state soddisfatte. Verranno inoltre esposti alcuni limiti della soluzione esposta e possibili spunti per migliorarla.

5.1 Obbiettivi raggiunti

Ricapitolando, la richiesta di questo lavoro consisteva nello sviluppo di un programma in grado di rilevare i movimenti involontari in immagini MRI. Tale applicativo sarebbe dovuto risultare di facile utilizzo, con un'interazione minima da parte dell'operatore, e veloce nello fornire una risposta al quesito se vi fosse stato o meno il movimento e dove. Entrambi questi obbiettivi sono stati raggiunti. Per quanto riguarda il primo, come già spiegato nella sezione Utilizzo del programma, il programma sviluppato non richiede alcun intervento da parte dell'operatore se non quello di fornire il `path` della cartella contenente i dati da analizzare. Per quanto concerne il secondo, l'utilizzo di meccanismi di *multithreading* e di funzioni altamente ottimizzate per l'elaborazione delle immagini hanno permesso di ottenere un programma con tempi di risposta minimi. In Figura 5.1 è riportato il tempo impiegato dal programma per analizzare uno dei casi reali del dataset, più precisamente il paziente 25.

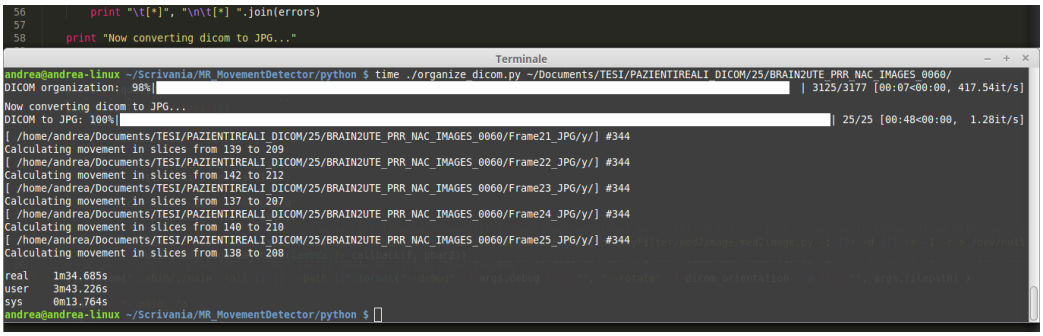


Figura 5.1: Output del programma con relativi tempi

Come è ben visibile in Figura 5.1, il tempo di esecuzione dell'intero processo risulta essere di soli 1 minuto, 34 secondi e 685 millesimi. Tale tempo è stato ottenuto eseguendo il programma su un computer con processore Intel i7 con 8GB di Random Access Memory (RAM) e sistema operativo Linux Mint 17.3. La quantità di dati analizzati risulta essere maggio di 3 mila file per un peso complessivo di oltre 1.2GB. Come già accennato, un tale successo è dovuto allo sfruttamento di tutti i processori a disposizione e all'aver convertito i dati grezzi presenti nei file DICOM in un formato più maneggevole e compresso quale è quello JPEG.

Questa trasformazione, però, se da una parte comporta grandi vantaggi in termini di velocità di esecuzione non è di certo esente da problematiche. Durante questa operazione, infatti, i livelli di grigio vengono rimappati passando da 16 o 17 bit di profondità a 8 bit. Si perdono sicuramente delle informazioni ma queste sono state considerate non fondamentali per la buona riuscita di questo progetto. Inoltre, sempre per quanto riguarda il tempo di esecuzione, la fase di conversione tra i due formati incide considerevolmente sul tempo totale.

5.2 Sviluppi futuri

Come si è appena visto, la fase di conversione tra il formato DICOM e quello JPEG comporta grandi vantaggi ma anche alcuni inconvenienti. Ragion per cui una possibile miglioria, nonché sviluppo futuro, potrebbe essere proprio quella di evitare questa trasformazione e analizzare direttamente i dati grezzi contenuti all'interno dei file DICOM. Durante lo studio di questa tesi

questa eventualità era stata presa in considerazione ma i benefici risultavano maggiori degli svantaggi.

Sempre sul fronte dell'efficienza un possibile sviluppo futuro potrebbe essere quello di sfruttare in maniera più profonda il *multithreading*. Questa tecnica consiste nel suddividere il carico di lavoro su *thread* diversi e velocizzarne quindi l'esecuzione. All'interno del progetto, come già spiegato, tale potente strumento è stato utilizzato durante la fase di trasformazione dei file DICOM a JPEG e durante le operazioni sulle immagini. Un'ulteriore miglioria potrebbe essere ottenuta sfruttando il *multithreading* durante l'analisi dei *frame*. Ora i *frame* vengono processati uno alla volta, in maniera sequenziale, ma, non essendoci alcun vincolo sull'ordine in cui devono essere analizzati, potrebbero essere elaborati in parallelo. In questo modo si risparmierebbe ulteriore tempo di esecuzione, unico vincolo sarebbe quello di attendere che tutti i *thread* finiscano il lavoro per valutare i risultati ottenuti.

Come ultimo sviluppo futuro, inteso come miglioria funzionale piuttosto che prettamente legato all'efficienza, si potrebbe cercare di applicare la stima della variazione dell'angolo, calcolata dal programma, per correggere il movimento. Questo processo, per i risultati che si sono riusciti ad ottenere, è stato soltanto implementato ma non utilizzato né testato. L'idea è quella di calcolare, per ogni vista di ogni *frame*, la variazione media dell'angolo di inclinazione e, nel caso che quel *frame* risulti mosso, applicare la variazione ad ogni rispettiva vista e ricostruire il volume DICOM. Nel seguente listato di codice Python è riportato parte del metodo che dovrebbe ricostruire il movimento individuato

```
reader = vtk.vtkDICOMImageReader()
reader.SetDirectoryName(args.dicom_folder)
reader.Update()

bounds = reader.GetOutput().GetBounds()

center = [ (bounds[1] + bounds[0]) * 0.5,
       (bounds[3] + bounds[2]) * 0.5,
       (bounds[5] + bounds[4]) * 0.5 ]

# Matrices for axial, coronal and sagittal
axial = vtk.vtkMatrix4x4()
axial.DeepCopy((1, 0, 0, center[0],
        0, 1, 0, center[1],
        0, 0, 1, center[2],
        0, 0, 0, 1))

coronal = vtk.vtkMatrix4x4()
coronal.DeepCopy((1, 0, 0, center[0],
          0, 0, 1, center[1],
          0,-1, 0, center[2],
          0, 0, 0, 1))
```

```
23
    sagittal = vtk.vtkMatrix4x4()
25  sagittal.DeepCopy((0, 0,-1, center[0],
                1, 0, 0, center[1],
27              0,-1, 0, center[2],
                0, 0, 0, 1))
29
    transform = vtk.vtkTransform()
31  # Move ref system origin to the center of the volume
    transform.Translate(center[0], center[1], center[2]);
33  # Apply rotation
    transform.RotateX( args.x );
35  transform.RotateY( args.y );
    transform.RotateZ( args.z );
37  # Move back system origin
    transform.Translate(-2*center[0], -2*center[1], -2*center[2]);
```

Acronimi

ADC Analog to Digital Converter. 19

API Application Programming Interface. 8, 12

CPU Central Process Unit. 8

CT Computed Tomography. 16

DFT Discrete Fourier Transform. 31, 32

DICOM Digital Imaging and Communications in Medicine. 4–6, 27–29, 45, 46, 51, 58, 59

DIMED Dipartimento di Medicina. 14

GPU Graphics Processing Unit. 10

JPEG Joint Photographic Experts Group. 28, 45–47, 58, 59

LPF Low Pass Filter. 31, 32

MR Magnetic Resonance. 1, 3, 4, 14, 16, 19, 29

MRI Magnetic Resonance Imaging. 3, 4, 57

NMR Nuclear Magnetic Resonance. 3, 4, 14–16, 51

OpenCV Open source Computer Vision library. 8–10, 12

PET Positron Emission Topography. 1, 3, 7

PET/CT Positron Emission Topography/Computed Tomography. 3

PET/MR Positron Emission Topography/Magnetic Resonance. 1, 2, 4

RAM Random Access Memory. 58

VR Value Representation. 6

Bibliografia

[1] J. P. Hornak, *The Basics of MRI*. 1996.

[2] J. W. Hennel and J. Klinowski, *Fundamentals of Nuclear Magnetic Resonance*. Longman Scientific & Technical, 1993.

[3] A. Kaehler and G. Bradski, *Learning OpenCV*. O'Reilly Media, 2015.

[4] C. H. Chen, *Computer Vision in Medical Imaging*. World Scientific Publishing Company, 2013.

[5] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

[6] P. Rinck, *Magnetic Resonance in Medicine.*, ch. 13. The Basic Textbook of the European Magnetic Resonance Forum., 9 ed., 2016.

[7] R. R. Fulton, S. R. Meikle, S. Eberl, J. Pfeiffer, and C. J. Constable, "Correction for head movements in positron emission tomography using an optical motion-tracking system," *IEEE Transactions on Nuclear Science*, vol. 49, no. 1, pp. 116–123, 2002.

[8] O. V. Olesen, J. M. Sullivan, T. Mulnix, R. R. Paulsen, L. Hojgaard, B. Roed, R. E. Carson, E. D. Morris, and R. Larsen, "List-mode pet motion correction using markerless head tracking: Proof-of-concept with scans of human subject," *IEEE Transactions on Medical Imaging*, vol. 32, pp. 200–209, Feb 2013.

[9] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998.

[10] P. J. Kostelec and S. Periaswamy, "Image registration for mri," *Modern Signal Processing*, vol. 46, pp. 161–184, 2003.

[11] G. T. Y. Chen and C. A. Pelizzari, "Image correlation techniques in radiation therapy planning," *Comput. Med. Imag. Graphics*, vol. 13, p. 235–240, 1989.

[12] G. T. Y. Chen, C. A. Pelizzari, D. R. Spelbring, R. R. Weichselbaum, and C. T. Chen, "Accurate three-dimensional registration of ct, pet and/or mr images of the brain," *Journal of Computer Assisted Tomography*, pp. 20–26, 1989.

[13] D. N. Levin, G. T. Y. Chen, C. A. Pelizzari, C. T. Chen, and M. D. Cooper, "Retrospective geometric correlation of mr, ct and pet images," *Radiology*, vol. 169, p. 817–823, 1988.

[14] S. W. Atlas, *Magnetic resonance imaging of the brain and spine*. Lippincott Williams & Wilkins, 4 ed., 2008.

[15] T. D. Cradduck and E. Busemann-Sokole, "Computers in nuclear medicine," *RadioGraphics*, vol. 5, no. 1, pp. 51–82, 1985.

[16] J. E. Koss, D. L. Kirch, E. P. Little, T. K. Johnson, and P. P. Steele, "Advantages of list-mode acquisition of dynamic cardiac data [ecg-gated nuclear medicine imaging]," *IEEE Transactions on Nuclear Science*, vol. 44, pp. 2431–2438, Dec 1997.

[17] P. "Gwosdek, S. Grewenig, A. Bruhn, and J. Weickert, *"Scale Space and Variational Methods in Computer Vision: Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29 – June 2, 2011, Revised Selected Papers"*, ch. "Theoretical Foundations of Gaussian Convolution by Extended Box Filtering". "Springer Berlin Heidelberg", "2012".

[18] K. Sreedhar and B. Panlal, "Enhancement of images using morphological transformations," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 4, Feb 2012.

[19] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, Nov 1986.

[20] M. Lutz, *Learning Python*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2 ed., 2003.

[21] B. K. P. Horn", ""understanding image intensities"," *"Artificial Intelligence"*, vol. "8", pp. "201–231", "1977".

[22] M. Petrou and C. Petrou, *Image Processing: The Fundamentals*. John Wiley & Sons, Ltd., 2010.

[23] S. Butterworth, "On the theory of filter amplifiers," *Experimental Wireless and the Radio Engineer*, vol. 7, pp. 536–541, 1930.

[24] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 32–46, 1985.

[25] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. Orlando, FL, USA: Academic Press, Inc., 2nd ed., 1982.

[26] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, 1962.

[27] M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing*. Butterworth-Heinemann, 2002.

[28] J. Kilian, "Simple image analysis by moments," 2001.