Università di Pisa

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
Corso di Laurea Specialistica in Scienze Fisiche

Tesi di Laurea Specialistica

Sviluppo di algoritmi per l'analisi automatica di immagini tomografiche polmonari

Candidato:
Francesco Bagagli

Relatore:
Dott.ssa
Maria Evelina Fantacci

Anno Accademico 2010–2011

Indice

Introduzione

Il lavoro oggetto di questa tesi consiste nello sviluppo di algoritmi per l'analisi automatica di immagini biomediche acquisite mediante Tomografia Assiale Computerizzata (TAC), con l'obiettivo di identificare le lesioni tumorali polmonari. In particolare sono stati sviluppati algoritmi paralleli, efficienti in termini di tempo di esecuzione, utilizzando le *Graphical Processing Units* (*GPU*) attraverso la tecnologia *CUDA* (*Compute Unified Device Architecture*) di NVIDIA.

Il tumore polmonare è attualmente la maggiore causa di morte fra le patologie neoplastiche e la TAC si è dimostrata essere la tecnica più efficace per la diagnosi precoce di questa patologia, il cui segno radiologico principale consiste nella formazione di noduli polmonari non calcifici. Recenti studi del National Lung Screening Trial (U.S.) hanno dimostrato che lo *screening* della popolazione asintomatica considerata a rischio, effettuato con la TAC spirale *multislice* toracica, è in grado di ridurre la mortalità del 20% rispetto a uno screening effettuato con la radiografia.

Un esame TAC, specialmente se acquisito nella modalità a bassa dose e strato sottile, come nel caso dello *screening*, è composto da molte (3-400) immagini bidimensionali molto rumorose che il radiologo deve analizzare attentamente alla ricerca di piccoli noduli. La grande mole di dati prodotta in una scansione TAC rende quindi la lettura delle immagini un compito lungo e gravoso per il radiologo. In letteratura è stato dimostrato che nella routine clinica molti noduli polmonari sono persi alla prima lettura.

Con queste premesse nasce l'esigenza di sviluppare uno strumento per affiancare il radiologo nel suo lavoro, che prende il nome di *Computer Aided Detection* (*CAD*).

In questo lavoro di tesi è descritto il sistema CAD per la ricerca automatica di noduli polmonari interni, sviluppato nell'ambito della collaborazione MAGIC-5 dell'INFN (Istituto Nazionale di Fisica Nucleare). Il CAD realizzato è costituito da tre moduli. Un primo modulo si occupa di isolare il volume polmonare dal resto dell'immagine utilizzando un processo basato su *thresholding*, *region-growing* e operatori morfologici. Successivamente un

secondo modulo si occupa di identificare i candidati noduli. Questi sono modellizzati come oggetti sferici di profilo gaussiano. La matrice tridimensionale dei dati viene analizzata con un algoritmo multi-scala basato sugli autovalori della matrice hessiana associata all'immagine, ideato per discriminare fra oggetti ad alta sfericità ed oggetti planari o cilindrici. Un ultimo modulo si occupa della riduzione del numero dei falsi positivi presenti fra i candidati noduli. Questo avviene caratterizzando ogni *voxel* (*volume pixel*) di ciascuna regione di interesse con dei vettori di *features* che vengono analizzati da un classificatore di tipo *Support Vector Machine*.

In un contesto di screening della popolazione in cui vengono prodotte molte immagini, la velocità di esecuzione del CAD può diventare un fattore limitante. Inoltre il CAD ha dimostrato di essere utile e funzionante anche applicato ad analisi cliniche, dove è importante la velocità di risposta del sistema.

Il lavoro descritto in questa tesi è consistito nell'utilizzo della tecnologia CUDA di NVIDIA per la programmazione su GPU con l'obiettivo di ridurre i tempi di esecuzione del CAD.

Negli ultimi anni l'incremento delle prestazioni dei software dovuta al semplice aumento della capacità di elaborazione dei microprocessori ha rallentato la sua crescita. Le applicazioni che invece continuano ad avere incrementi di prestazioni sono i programmi paralleli, con i quali più processi cooperano a completare il lavoro in modo più rapido.

L'industria dei semiconduttori si è quindi orientata verso lo sviluppo di due principali paradigmi di processori: le *multi-core* CPU (Central Processing Unit) e le *many-core* GPU (Graphical Processing Unit). Queste ultime sono progettate per eseguire computazioni intense e altamente parallele, e sono particolarmente adatte a risolvere problemi che possono essere espressi come calcoli paralleli sui dati: il dato viene partizionato e lo stesso programma viene eseguito in parallelo su tutti gli elementi.

In particolare in questo lavoro di tesi è stata studiata e realizzata una implementazione parallela dell'algoritmo di identificazione dei candidati noduli.

Nel primo capitolo è descritto l'ambito del problema: le problematiche legate allo screening per la diagnosi precoce del tumore polmonare, le caratteristiche delle immagini TAC, cosa è un sistema CAD, i criteri per valutare le sue prestazioni e i dati disponibili. La descrizione del sistema CAD realizzato a Pisa, con in particolare l'algoritmo realizzato per l'identificazione dei candidati noduli interni, è presentato nel secondo capitolo. Nel terzo capitolo sono descritte le GPU e l'architettura CUDA. L'implementazione dell'algoritmo di cui sopra, basata sull'architettura CUDA, è descritta nel quarto. I risultati ottenuti in termini di guadagno sul tempo di esecuzione e il confronto fra le

due implementazioni del sistema CAD realizzate, sono riportati nel quinto capitolo. Infine sono tratte le conclusioni e le possibili prospettive del lavoro effettuato.

Capitolo 1

Diagnosi precoce del tumore polmonare e *Computer Aided Detection*

1.1 Cosa è un sistema CAD

Le moderne tecniche di generazione di immagini biomediche digitali a scopo diagnostico, come la radiografia digitale ma soprattutto la risonanza magnetica e la Tomografia Assiale Computerizzata (TAC), generano una grande mole di informazioni che i radiologi devono analizzare e valutare, anche in brevi periodi di tempo. In questo contesto un aiuto può essere costituito da sistemi di analisi automatica delle immagini. Nel caso di immagini tomografiche il radiologo deve ricostruire mentalmente la struttura tridimensionale degli oggetti analizzando immagini di sezioni bidimensionali in cui le proiezioni delle strutture patologiche si confondono facilmente con le proiezioni delle normali strutture anatomiche. I sistemi di Computer Aided Detection (CAD) possono quindi assistere il radiologo nell'interpretazione dei dati provenienti da immagini mediche, evidenziando in modo automatico zone con un sospetto di patologia.

Lo sviluppo di sistemi CAD è una disciplina relativamente nuova e multidisciplinare che combina elementi di intelligenza artificiale con l'analisi di immagini digitali e radiologiche. Molti ospedali utilizzano già in via sperimentale sistemi CAD, ad esempio nei programmi di screening della popolazione per la diagnosi del cancro al seno tramite mammografia e per l'identificazione di polipi nel colon con la colonscopia virtuale.

1.2 Descrizione del problema

Il cancro ai polmoni è il tipo di cancro più diffuso nel mondo, e tra le cause di morte per cancro negli Stati Uniti si attesta al 28% del totale, nell'Unione Europea al 19%. Si manifesta principalmente sotto forma di noduli polmonari non calcifici: allo stadio iniziale della malattia è essenzialmente asintomatico. Nonostante i progressi nel campo della chirurgia, della radioterapia e della chemioterapia il tasso di sopravvivenza medio a 5 anni per questa malattia rimane basso [1]. Nel caso di diagnosi effettuata nello stadio avanzato della malattia tale tasso di sopravvivenza è del 16%, mentre nel caso di diagnosi precoce questo valore si alza fino al 53%. Solo nel 15% dei casi però la diagnosi è effettuata ad uno stadio precoce della malattia [2].

Per il riconoscimento di piccoli noduli polmonari la Tomografia Assiale Computerizzata (TAC) si è dimostrata essere la tecnica di imaging più sensibile, in particolare dopo l'introduzione della TAC spirale [3]. È stato dimostrato che la TAC spirale a bassa dose identifica più noduli e tumori allo stadio iniziale rispetto alla radiografia toracica [4].

Vari programmi di screening della popolazione sono in corso per capire l'efficacia della diagnosi precoce effettuata mediante TAC nella riduzione della mortalità. I primi risultati positivi sono stati ottenuti dallo *screening trial* statunitense organizzato dal NLST (National Lung Screening Cancer) [5]. Questo studio mette a confronto i due metodi utilizzati per effettuare diagnosi durante i programmi di screening: uno è la TAC, l'altro è la radiografia toracica. I risultati pubblicati indicano una riduzione del 20% della mortalità per cancro fra i partecipanti sottoposti a TAC, rispetto a quelli sottoposti a radiografia toracica [6].

La mole di dati generati da un esame TAC, soprattutto se effettuato con macchinari di ultima generazione, può essere molto grande: per un singolo caso il radiologo si trova a dover analizzare da 300 a 500 fette bidimensionali contenenti tipicamente 512×512 voxels. Queste immagini sono tipicamente più rumorose rispetto a quelle ottenute con una dose standard e, a seconda del protocollo di screening, il radiologo deve individuare anche noduli molto piccoli. È stato inoltre dimostrato che un gran numero di noduli (20-35%) può essere non rilevato durante la prima refertazione [7]. Per supportare il radiologo durante la diagnosi quindi, possono essere utilizzati sistemi CAD.

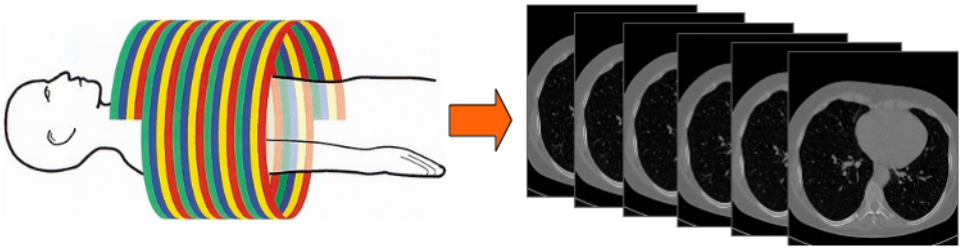


Figura 1.1: Esempio di dati ricostruiti dopo una scansione TAC toracica: immagini bidimensionali rappresentanti sezioni trasversali del torace.

1.3 Materiali e metodi

1.3.1 Tomografia Assiale Computerizzata

La tecnica della Tomografia Assiale Computerizzata utilizza gli stessi principi fisici della radiografia (interazione e attenuazione di un fascio di raggi X con la materia) per generare una serie di sezioni assiali dell'interno del corpo umano (figura 1.1 e 1.4).

La tecnica più utilizzata attualmente è la TAC spirale multi-*slice*. È costituita da un sistema composto da un tubo radiogeno ruotante accoppiato ad un sistema di rivelatori posti circolarmente (nel caso della multi-slice su più piani contigui) intorno all'oggetto da analizzare. Il tubo ruota in modo continuo attorno al paziente, mentre quest'ultimo trasla all'interno della struttura (1.2): nel sistema di riferimento solidale con il lettino il tubo radiogeno esegue una traiettoria spirale, da cui il nome.

Dai dati proiettivi acquisiti, un algoritmo di ricostruzione permette di ricostruire le sezioni assiali nel piano xy. Ogni pixel dell'immagine ricostruita si riferisce ad un elemento di volume, *voxel* (*volume pixel*), e contiene un numero legato al coefficiente di attenuazione corrispondente al tessuto contenuto all'elemento di volume considerato. Si utilizzano soprattutto le unità di Hounsfield (HU). Il valore in HU di un determinato materiale è definito come

$$HU = 1000\,\frac{\mu_{tessuto} - \mu_{acqua}}{\mu_{acqua}} \tag{1.1}$$

dove $\mu_{tessuto}$ è il coefficiente di attenuazione lineare medio corrispondente al voxel, e μ_{acqua} è il coefficiente di attenuazione dell'acqua.

Poichè i coefficienti di attenuazione μ dipendono dall'energia dei fotoni incidenti e dal tipo di materiale attraversato, e dato che il tubo radiogeno non produce un fascio di raggi X monocromatico (figura 1.3), il valore di HU di ogni tessuto è di fatto una media delle HU ottenute ad ogni energia

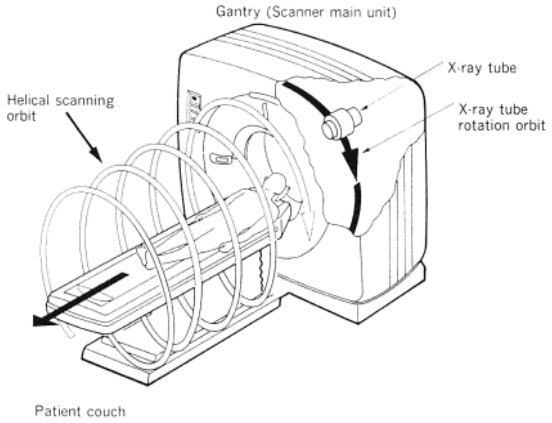


Figura 1.2: Schema raffigurante il funzionamento di una TAC elicoidale. È evidenziato il movimento del tubo radiogeno (*X-ray tube*) lungo la sua orbita (*rotation orbit*) all'interno del macchinario (*gantry*). Il tavolo su cui è posto il paziente (*patient couch*) trasla all'interno del macchinario: nel sistema di riferimento solidale con il tavolo, il tubo descrive un'orbita spirale (*helical scanning orbit*).

Tabella 1.1: Valori di HU per alcuni tessuti presenti in una TAC toracica.

materiale	HU
aria	-1000
grasso	-120
acqua	0
muscolo	40
ossa	$\gtrsim$400

presente nello spettro. In tabella 1.1 sono riportati alcuni valori delle HU per differenti tessuti.

I principali parametri che caratterizzano una scansione TAC sono la tensione dell'alimentazione del tubo radiogeno (espressa in kVp), legata all'energia del fascio e l'esposizione, data dal prodotto fra la corrente del tubo radiogeno ed il tempo (espressa in $mA \cdot s$), collegata al numero totale di fotoni emessi.

In figura 1.4 è riportata una immagine TAC toracica, in cui sono evidenziate alcune strutture presenti.

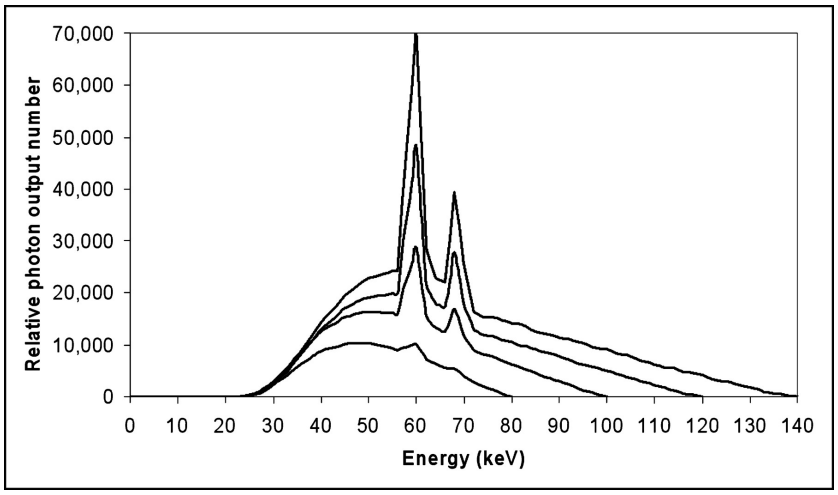


Figura 1.3: Spettro di emissione per un tubo radiogeno con anodo di tungsteno. La massima energia dei fotoni è determinata dal voltaggio del tubo. Sono riportati i voltaggi di 80, 100, 120 e 140 kVp [8].

1.3.2 Dati utilizzati

Nello sviluppo di un sistema CAD sono fondamentali due requisiti: disponibilità di un database di immagini e di un insieme di annotazioni, da considerare come verità di riferimento per lo sviluppo del sistema e l'allenamento degli algoritmi [9]. Ottenere questi dati è molto costoso in termini di fondi e di tempo, soprattutto per quanto riguarda la disponibilità di annotazioni realizzate da radiologi qualificati. Inoltre il rischio è che ogni gruppo di ricerca utilizzi un proprio database, rendendo difficile la comparazione dei vari sistemi CAD. Per questo lavoro di tesi sono stati utilizzati database pubblici di ricerca.

LIDC Database

The Lung Image Database Consortium (LIDC) [10] è un database multicentrico (US), attualmente composto da più di 1000 scansioni [11], ottenute con differenti macchinari e parametri di acquisizione e ricostruzione (kVp, corrente del tubo radiogeno, collimazione e spessore ricostruito). Questa variabilità rappresenta quindi una tipica situazione reale in un programma di screening multicentrico, e lo rende un database utile per lo sviluppo di sistemi CAD.

Poichè, a differenza di altre patologie, gli approfondimenti diagnostici successivi alla TAC sono costituiti da esami potenzialmente invasivi o comunque pericolosi (biopsia polmonare) o effettuabili solo per noduli di notevoli

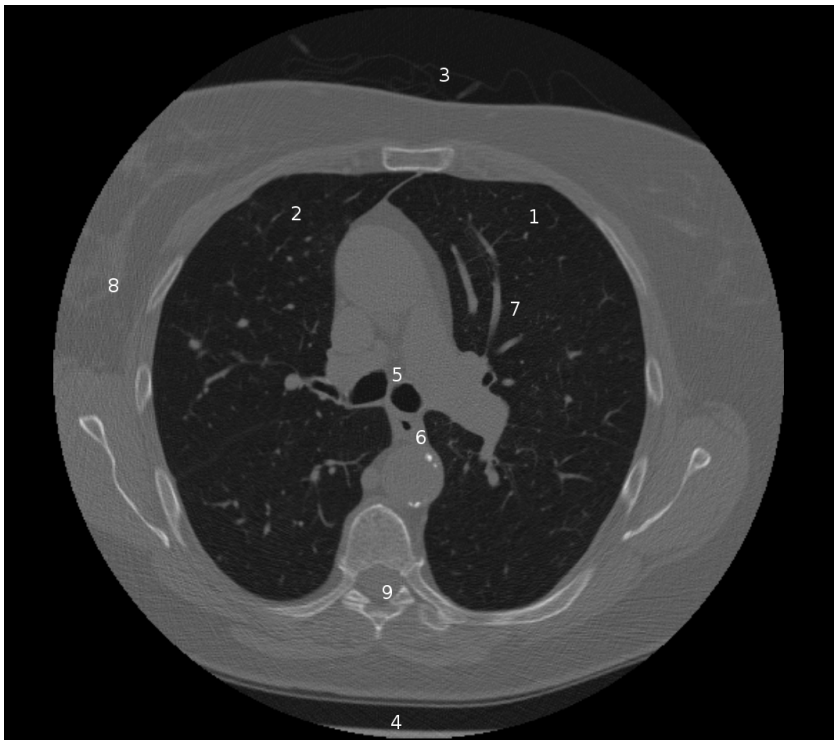


Figura 1.4: Esempio di immagine TAC del torace umano. Si possono identificare le seguenti strutture: polmone sinistro (1), polmone destro (2), aria al di fuori del corpo (3), lettino su cui è sdraiato il paziente (4), trachea (5), esofago (6), vasi sanguigni (7), tessuto adiposo (8), spina dorsale (9).

dimensioni ($> 0.8 - 1\,mm$ per la PET), non esiste un *gold-standard* oggettivo per la diagnosi. Il consorzio LIDC ha scelto di mettere a disposizione per ogni scansione 4 annotazioni ottenute da 4 differenti radiologi esperti, non obbligandoli a giungere a un *consensus* dell'annotazione, il cui risultato tende a riflettere l'opinione del membro più autorevole [10].

Avendo quindi a disposizione le varie annotazioni, ottenute con una lettura in due fasi, la prima *blinded* e la seconda *unblinded*, si possono costruire i *gold-standard* partendo dalle annotazioni comuni a uno, due o più radiologi a seconda dei requisiti richiesti in termini di compromesso fra sensibilità raggiungibile e numero di falsi positivi accettabili.

Le annotazioni a disposizione [12] contengono sia i noduli identificati come tali, sia alcuni oggetti simili ai noduli con diametro > 3 mm che però appartengono alla classe dei falsi positivi.

Fra tutte le scansioni disponibili sono state selezionate 138 scansioni a "fetta sottile", ovvero con spessore della fetta ricostruita compresa nell'inter-

vallo $0.5 - 2.0$ mm. Infatti gli oggetti ricercati in un programma di screening e dal CAD sono principalmente i noduli di piccole dimensioni. È necessario quindi che lo spessore della fetta ricostruita sia più piccola delle dimensioni caratteristiche di questi noduli.

Le scansioni utilizzate in questa analisi sono state acquisite a 120 kVp, con una corrente variabile fra i 40 e i 172 mA. Il *voxel-spacing*, ovvero la dimensione fisica in mm rappresentata da 1 voxel dell'immagine, lungo nelle direzioni ortogonali all'asse del paziente è compreso nell'intervallo $0.434 - 0.762\,mm$. Il numero di fette per ciascuna scansione varia da 154 a 730.

Tutte le immagini sono memorizzate nel formato DICOM (*Digital Imaging and Communications in Medicine*) [13], di comune utilizzo in medicina. In questo formato sono memorizzati non solo i dati propri dell'immagine ma anche un insieme di metadati con molte informazioni aggiuntive: parametri di acquisizione, spessore della fetta ricostruita, etc.

1.3.3 Figure di merito di un sistema CAD

Per confrontare le capacità di differenti sistemi CAD è necessario utilizzare una figura di merito che ne evidenzi le prestazioni.

Per questo scopo si utilizza la metodologia Free-Response Receiver Operating Characteristic (FROC).

L'output di un sistema CAD è tipicamente una lista di candidati noduli, ciascuno dei quali è corredato da un insieme di coordinate e da un valore scalare $P_{patologia}$ corrispondente al livello di confidenza che l'oggetto identificato sia un vero nodulo.

Scegliendo un valore di soglia t è possibile selezionare solo i noduli il cui livello di confidenza sia maggiore della soglia scelta ($P_{patologia} > t$). Definendo un criterio di prossimità fra annotazioni e candidati noduli (tipicamente 1.5 volte il raggio contenuto nelle annotazioni dei radiologi) è possibile identificare quanti di questi oggetti identificati dal CAD corrispondano a lesioni vere o siano Falsi Positivi (FP). Per ogni valore di soglia (t) si ottiene una coppia di valori: la *sensibilità*, ovvero il rapporto fra il numero di noduli identificati dal CAD rispetto al numero totale di noduli identificati dal radiologo, e il numero di FP indicati dal sistema come noduli.

Al variare della soglia t si ottengono i punti che costituiscono la curva FROC del sistema. Un esempio di curva è riportato in figura 1.5.

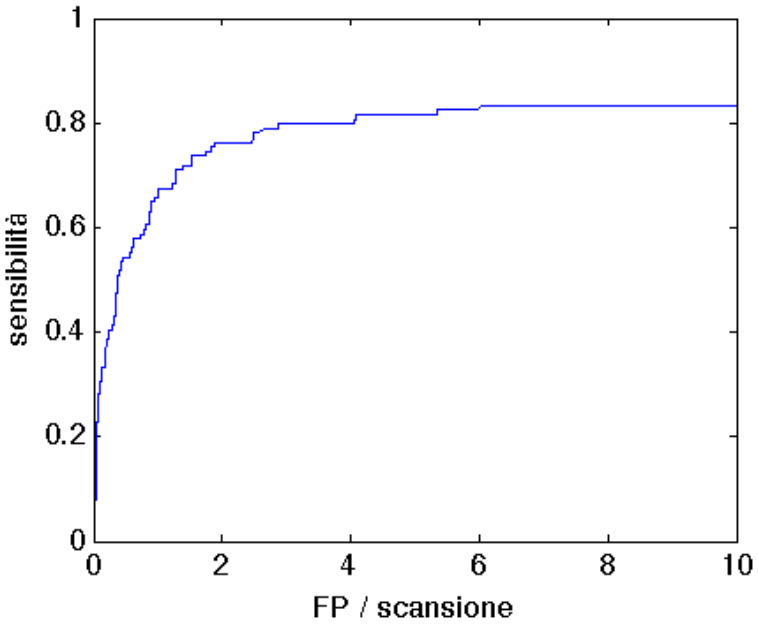


Figura 1.5: Tipica curva FROC di un sistema: in ascissa sono riportati il numero di falsi positivi per scansione, mentre in ordinata è riportata la sensibilità.

Capitolo 2

Descrizione del sistema CAD

2.1 I noduli polmonari

Il sistema CAD presentato ha come obiettivo la ricerca di noduli polmonari non calcifici presenti nelle TAC toraciche. I noduli polmonari sono oggetti non più grandi di 3 cm di diametro che nella TAC assumono l'aspetto di opacità circolari [9].

Per poter sviluppare un sistema CAD è necessaria una conoscenza di base delle principali strutture anatomiche presenti nel torace. Alcune di queste possono essere confuse con strutture nodulari quando vengono visualizzate in proiezioni assiali bidimensionali.

I noduli polmonari identificabili nelle scansioni TAC sono stati suddivisi in due classi principali, in base alla loro posizione all'interno del volume polmonare e alla loro morfologia:

- noduli interni: oggetti di forma sferica, interamente immersi nel parenchima polmonare (figura 2.1)
- noduli pleurici: oggetti di forma semisferica connessi alla parete pleurica (figura 2.2)

La differenza morfologica degli oggetti in queste due classi ha portato allo sviluppo di due sistemi CAD: CAD_I per l'identificazione dei noduli interni, e CAD_{JP} per l'identificazione dei noduli pleurici.

2.2 Il sistema CAD-Pisa

Entrambi i sistemi CAD si sviluppano in tre fasi. In primo luogo viene identificato il volume polmonare all'interno della matrice dei dati TAC. Successivamente, un algoritmo specifico per le due tipologie di nodulo, identifica

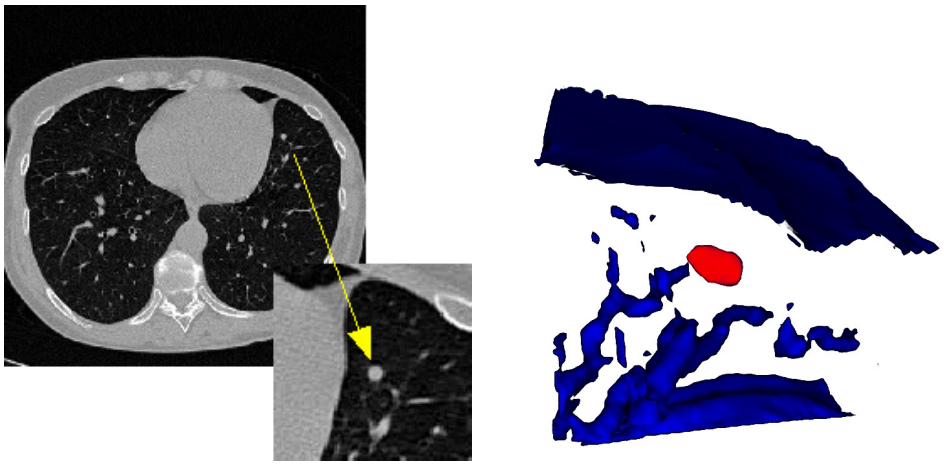


Figura 2.1: Esempio di nodulo interno: (a) nodulo interno come appare su una immagine TAC, (b) *rendering* tridimensionale.

una lista di Regioni di Interesse (ROI) contenenti le posizioni dei candidati noduli: *ROI Hunter* per i noduli interni (RH_I) e *ROI Hunter* per i noduli pleurici (RH_{JP}). Infine, le liste di candidati noduli sono classificate con una Support Vector Machine (SVM) lineare, con l'obiettivo di ridurre il numero dei falsi positivi identificati.

Il diagramma di flusso dell'analisi di una scansione TAC è mostrato in figura 2.3.

2.2.1 *Preprocessing* dei dati

Gli algoritmi di ricerca di forme sferiche e semisferiche utilizzati nei moduli RH_I e RH_{JP} necessitano, per avere le migliori prestazioni, di matrici di dati isotropiche, in cui ogni voxel abbia lo stesso passo spaziale (*voxel spacing*, v) nelle tre direzioni principali. Le immagini TAC, invece, sono tipicamente anisotrope: $v_x = v_y$ e $v_z \neq v_y$.

Per rendere isotropica l'immagine è stato eseguito uno *smoothing* lineare seguito da un ricampionamento dell'immagine tramite un processo di interpolazione lineare.

2.2.2 Segmentazione del volume polmonare

Prima di eseguire gli algoritmi di ricerca dei noduli, è necessario individuare la regione dell'immagine TAC che contiene il volume polmonare. Questo procedimento in analisi delle immagini è chiamato *segmentazione*. Il processo

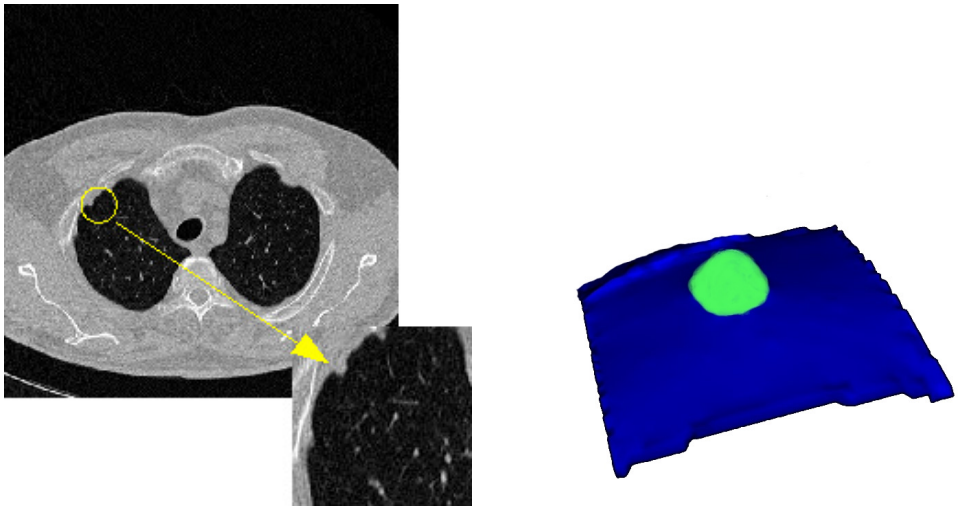


Figura 2.2: Esempio di nodulo pleurico: (a) nodulo pleurico come appare su una immagine TAC, (b) *rendering* tridimensionale.

di segmentazione si riferisce alla suddivisione di un'immagine digitale in più parti (insiemi di pixel) corrispondenti agli elementi presenti nell'immagine, ad esempio selezionando le varie strutture anatomiche. L'obiettivo di questa operazione è semplificare la rappresentazione dell'immagine, rendendola più semplice da analizzare.

Per il sistema CAD l'identificazione del volume polmonare è importante per evitare di segnalare ROIs all'esterno del polmone. Inoltre, una corretta ricostruzione della superficie polmonare esterna è necessaria al modulo di ricerca dei noduli pleurici.

Il metodo utilizzato per la procedura di segmentazione si ispira a quello presentato in [14]. Dall'immagine originale (figura 2.4a) vengono estratti i voxel di bassa intensità utilizzando una procedura di *thresholding* ottenendo una maschera binaria (figura 2.4b). I voxel appartenenti ad un determinato intervallo di intensità vengono etichettati come *foreground*, gli altri come *background*. L'intervallo utilizzato per determinare i voxel di *foreground* è dato da $T_{low} = -1000\,HU$ e $T_{high} = -600\,HU$.

La maschera binaria ottenuta è analizzata ricercando le varie componenti connesse costituite dai voxel di *foreground*. La componente connessa di dimensione maggiore, il cui bordo non tocca gli estremi dell'immagine, contiene i voxel appartenenti all'interno del polmone (figura 2.4c).

I vasi e i bronchi interni al polmone in questa procedura non risultano inclusi nella maschera. Per includerli senza modificare la morfologia della

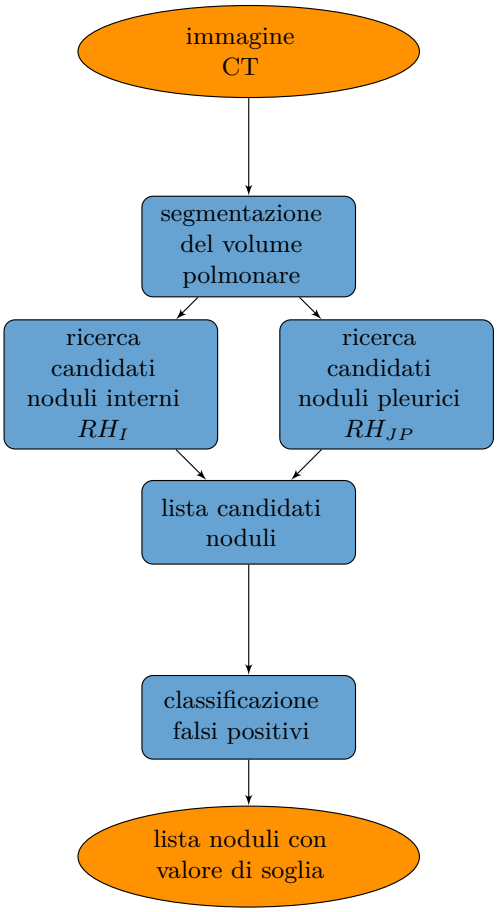


Figura 2.3: Diagramma del flusso di elaborazione dei dati da parte del CAD.

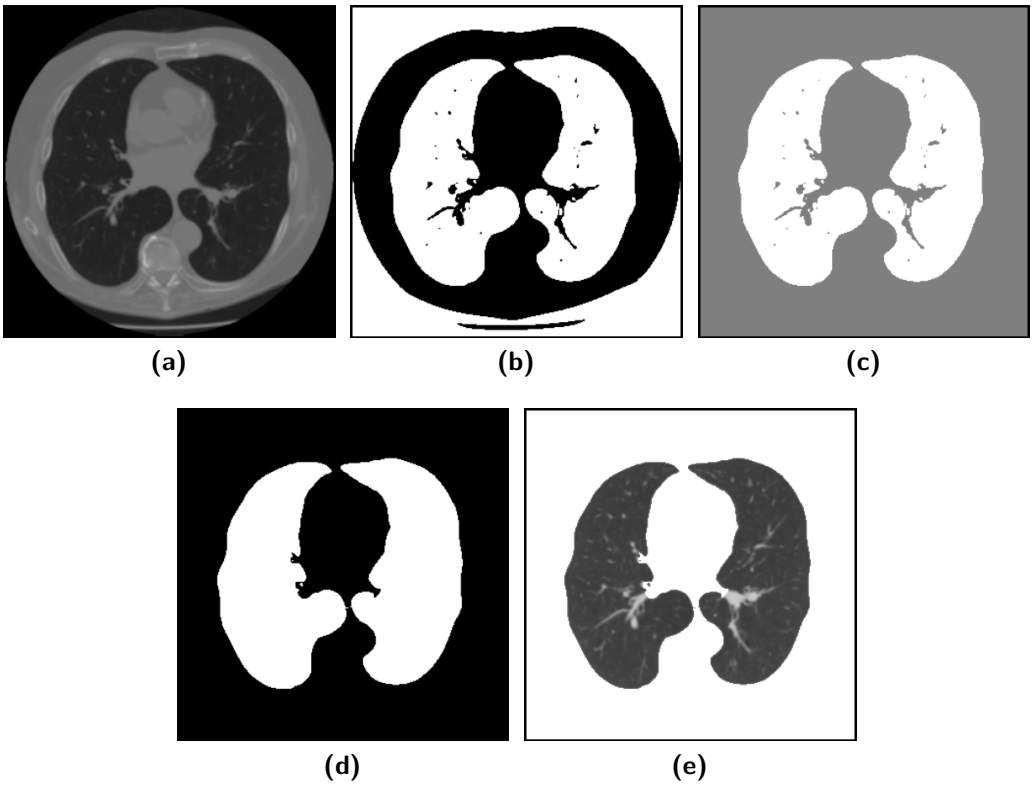
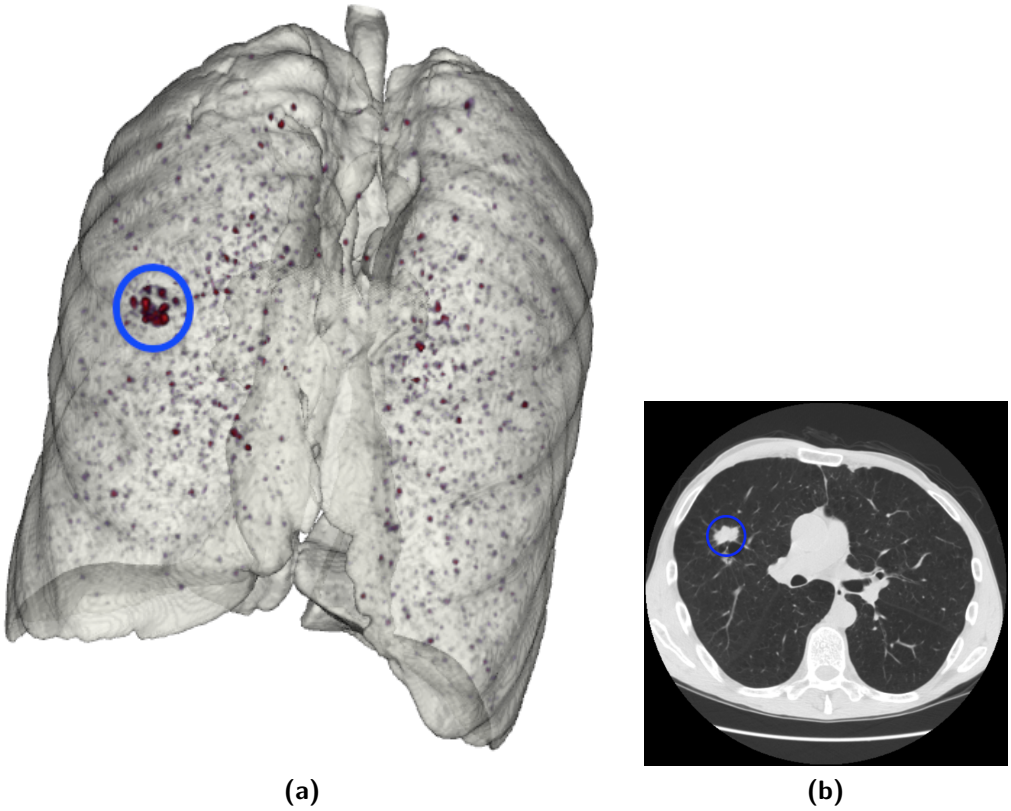


Figura 2.4: Segmentazione del volume polmonare: (a) immagine originale, (b) risultato dell'operazione di *thresholding*, (c) componente connessa più grande che non tocca il bordo, (d) rimozione dei vasi e dei bronchi, (e) dati originali sovrapposti alla maschera binaria ottenuta.

superficie pleurica, e quindi la forma dei noduli pleurici, è stata applicata una combinazione di operatori morfologici [15].

L'operatore di dilatazione e, succesivamente, quello di erosione sono stati applicati alla maschera binaria associata alla componente connessa individuata. Con una procedura di OR logico voxel per voxel, fra la maschera appena ottenuta e quella relativa al processo di thresholding, è stata ottenuta la maschera relativa al volume polmonare cercato (figura 2.4d). I raggi del supporto sferico utilizzato nella procedura di dilatazione (r_d) ed erosione (r_e) sono stati scelti in base a considerazioni anatomiche: per riempire cavità la cui dimensione è approssimativamente uguale a quella dei bronchi è necessario un raggio r_d di circa $8\,mm$, mentre il parametro r_e è stato scelto in modo da non modificare la forma dei noduli pleurici a causa del processo di dilatazione ($r_e = 16\,mm$).



(a) (b)

Figura 2.5: (a) Visualizzazione tridimensionale del volume polmonare, a cui è sovrapposto (in rosso) il punteggio del modulo RH_I. Nel cerchio blu è evidenziato un nodulo interno. In (b) è riportato lo stesso nodulo, evidenziato in una immagine bidimensionale.

2.2.3 Individuazione dei candidati noduli interni

I noduli interni possono essere modellizzati efficacemente come oggetti sferici di profilo gaussiano. Per identificare questo tipo di oggetti la matrice tridimensionale dei dati viene analizzata con un algoritmo multi scala basato sugli autovalori della matrice hessiana associata all'immagine [16], ideato per discriminare oggetti ad alta sfericità da oggetti planari o cilindrici.

Ciascun voxel della matrice risultante, $Z_{max}(i, j, k)$, contiene un valore scalare corrispondente alla probabilità di essere centro di una struttura sferica (figura 2.5).

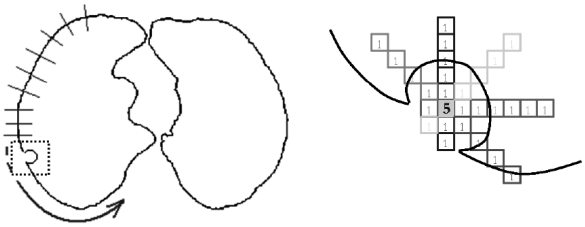


Figura 2.6: Descrizione del funzionamento dell'algoritmo di identificazione dei noduli pleurici. Ogni voxel accumula un punteggio proporzionale al numero di rette che si intersecano in quel punto.

2.2.4 Individuazione dei noduli pleurici

I noduli pleurici possono essere modellizzati come oggetti concavi semisferici, connessi alla pleura. Per identificare questo tipo di oggetti è stato implementato un metodo ispirato da [17] e basato sulla sovrapposizione delle normali alla superficie pleurica (*Pleura Surface Normal* - PSN) [18, 19]. Questo approccio consiste nella ricerca dei punti in cui più normali si intersecano (figura 2.6 e 2.7). Questi punti sono tipicamente centri di regioni convesse ad alta curvatura.

L'output di questo algoritmo è una matrice $A(i, j, k)$, i cui voxel contengono un valore proporzionale al numero di rette che si incrociano in quel punto.

Questa matrice viene poi analizzata alla ricerca di massimi locali corrispondenti alle posizioni dei candidati noduli pleurici.

2.2.5 Classificazione

Le liste dei candidati noduli prodotte dagli algoritmi descritti nelle sezioni precedenti (2.2.3 e 2.2.4) contengono, oltre ai veri noduli, anche un gran numero di falsi positivi. Il numero dei falsi positivi deve essere ridotto per migliorare le prestazioni del sistema CAD.

Inizialmente le liste di candidati noduli, ottenute dall'analisi dei massimi locali delle matrici dei punteggi Z_{max} e A, vengono ordinate in base ai valori di questi massimi locali. Una prima riduzione del numero dei falsi positivi avviene escludendo quei candidati i cui valori sono sotto una soglia minima prefissata.

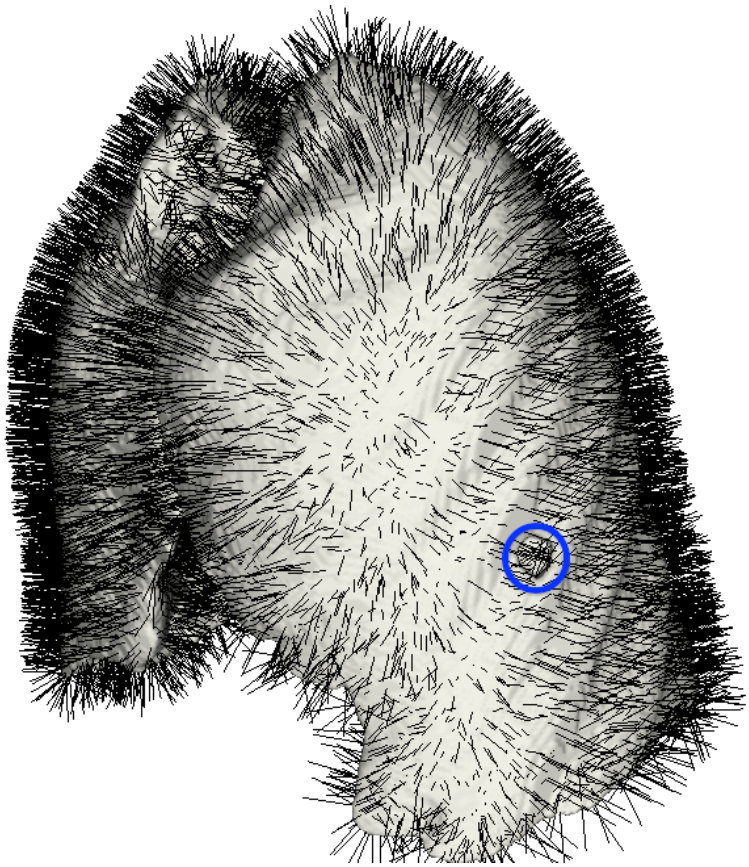


Figura 2.7: Visualizzazione delle normali calcolate sulla superficie pleurica. Nel cerchio blu è evidenziato un nodulo pleurico.

Per poter costruire un classificatore è cruciale riuscire ad esprimere l'informazione relativa all'oggetto da classificare in modo sintetico, utilizzando insiemi di valori scalari, chiamati *features*.

Per generare questi insiemi di features è stato utilizzato il metodo Voxel Based Neural Approach (VBNA) [19]. Ad ogni candidato nodulo viene associata una ROI, e per ciascun voxel appartenente a questa ROI viene calcolato un vettore di features.

La ROI associata a ciascun candidato nodulo viene identificata con una procedura basata su *thresholding*. Data una sfera S di 5 mm attorno al candidato nodulo vengono selezionati i voxel v tali che $I_v > t$ dove I_v è il valore di intensità del voxel e t è una soglia adattiva definita come

$$t = \max_{v \in S} I_v - \frac{1}{3}\left(\max_{v \in S} I_v - \min_{v \in S} I_v\right) \tag{2.1}$$

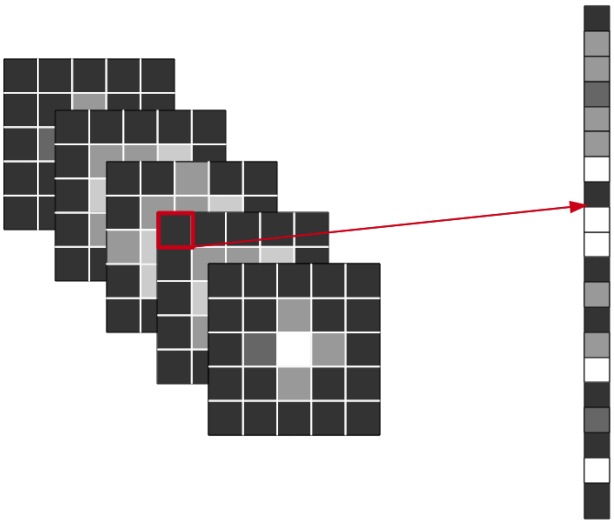


Figura 2.8: Creazione del vettore di features: ciascun voxel analizzato viene aggiunto al vettore.

Successivamente ad ogni voxel selezionato viene associato un vettore di features composto di due parti:

- la prima parte comprende i valori di intensità dei voxel appartenenti ad una regione cubica di lato 5 voxel centrata sul voxel analizzato. Questo intorno viene letto in sequenza lineare e costituisce la prima parte del vettore (vedi figura 2.8).

- a questi si aggiungono i tre autovalori della matrice gradiente:

$$G_{i,j} = \sum \partial_{x_i} I \partial_{x_j} I \quad \text{con } i,j{=}1,\ldots,3 \tag{2.2}$$

  (dove la somma è estesa ai primi vicini del voxel analizzato), e i tre autovalori della matrice hessiana:

$$H_{i,j} = \partial^2_{x_i x_j} I \quad \text{con } i,j{=}1,\ldots,3 \tag{2.3}$$

  dove $I(x_1, x_2, x_3)$ è l'immagine.

Ad ogni voxel della ROI è quindi associato un vettore di features costituito da 131 elementi.

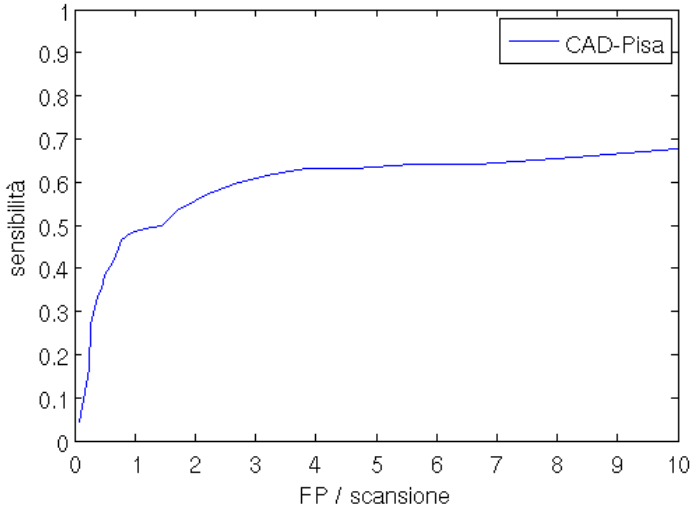


Figura 2.9: Curva FROC del sistema CAD-Pisa [19] sul database LIDC.

Con questi vettori di features viene allenato un classificatore di tipo Support Vector Machine (SVM) lineare[1].

La SVM è allenata ad assegnare i voxel in due classi: tessuto normale e tessuto nodulare.

Alla fine della procedura, ad ogni candidato nodulo è associato un valore di nodularità dato dalla media dei valori ottenuti per ciascun voxel appartenente alla ROI associata.

2.3 Risultati

Questo sistema CAD è stato allenato e validato su vari dataset e i risultati sono stati pubblicati in [22, 23, 19].

[1]Le Support Vector Machines (SVM) [20, 21] sono un insieme di metodi di apprendimento supervisionato utilizzati tipicamente per la classificazione di dati o in analisi di regressione. Il paradigma delle SVM non è basato su una analogia biologica, come nel caso delle reti neurali artificiali, ma sulla ricerca di iperpiani che separano i dati con il più largo margine possibile. In questo lavoro sono state utilizzate esclusivamente SVM lineari utilizzate per eseguire una classificazione binaria. Dato un insieme di esempi di allenamento, ciascuno costituito da un vettore di dati ($\mathbf{x_i}$) e dalla classe di appartenenza ($d_i = \pm 1$), l'algoritmo di allenamento SVM identifica l'iperpiano con il miglior margine di separazione fra le due classi. Nuovi esempi sono successivamente classificati calcolando la loro distanza con segno dall'iperpiano ottimale.

In figura 2.9 è riportata la curva FROC del sistema CAD-Pisa appena descritto sul database LIDC [19]. Il CAD ottiene il 63% di sensibilità a 3.8 FP/scansione, valori competitivi con quelli presenti in letteratura [19, 24].

2.4 *Roi Hunter* dei noduli interni

In questo paragrafo è descritto in dettaglio l'algoritmo utilizzato per l'identificazione dei candidati noduli interni, poichè è l'oggetto studiato in questo lavoro di tesi.

Nel CAD_I i noduli interni ricercati sono modellizzati come strutture sferiche. L'identificazione delle regioni di interesse viene effettuata con un algoritmo chiamato *dot-enhancer* [16], basato sul riconoscimento delle forme sferiche mediante l'analisi degli autovalori della matrice hessiana. Nel paragrafo 4.1 saranno descritte le implementazioni tecniche effettuate per ridurre la complessità computazionale dell'algoritmo.

2.4.1 Algoritmo *dot-enhancer*

L'algoritmo presentato in [16] è stato ideato per discriminare nell'immagine TAC le strutture sferiche nodulari dalle strutture anatomiche normalmente presenti all'interno del volume polmonare, come vasi o pareti. L'algoritmo sviluppato è progettato per avere una forte risposta a strutture sferiche (*sensibilità*), e una buona soppressione delle strutture planari o cilindriche (*specificità*). Con un analogo procedimento sarebbe possibile sviluppare algoritmi per l'enfasi di strutture cilindriche o planari. Per lo sviluppo sono state utilizzate le seguenti strutture ideali:

$$d(x,y,z) = exp\left(-\frac{x^2+y^2+z^2}{2\sigma^2}\right) \tag{2.4}$$

$$l(x,y,z) = exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.5}$$

$$p(x,y,z) = exp\left(-\frac{x^2}{2\sigma^2}\right) = G(x,\sigma) \tag{2.6}$$

corrispondenti rispettivamente a una sfera, un cilindro e un piano a sezione gaussiana di larghezza σ. Utilizzando i polinomi di Hermite $H_n(x)$:

$$\frac{\partial^n G(x,\sigma)}{\partial x^n} = (-1)^n \cdot \frac{1}{(\sigma\sqrt{2})^2} \cdot H_n\left(\frac{x}{\sigma\sqrt{2}}\right) \cdot G(x,\sigma) \tag{2.7}$$

e

$$H_0(x) = 1 \qquad H_1(x) = 2x \qquad H_2(x) = 4x^2 - 2$$

è possibile esprimere la matrice Hessiana (H_{dot}) per la sfera $d(x, y, z)$:

$$H_{dot}(x, y, z) = \frac{G(x, y, z, \sigma)}{2\sigma^2} \mathbb{H} \tag{2.8}$$

dove:

$$\mathbb{H} = \begin{pmatrix} H_2(\frac{x}{\sqrt{2}\sigma}) & H_1(\frac{x}{\sqrt{2}\sigma})H_1(\frac{y}{\sqrt{2}\sigma}) & H_1(\frac{x}{\sqrt{2}\sigma})H_1(\frac{z}{\sqrt{2}\sigma}) \\ H_1(\frac{y}{\sqrt{2}\sigma})H_1(\frac{x}{\sqrt{2}\sigma}) & H_2(\frac{y}{\sqrt{2}\sigma}) & H_1(\frac{y}{\sqrt{2}\sigma})H_1(\frac{z}{\sqrt{2}\sigma}) \\ H_1(\frac{z}{\sqrt{2}\sigma})H_1(\frac{x}{\sqrt{2}\sigma}) & H_1(\frac{z}{\sqrt{2}\sigma})H_1(\frac{y}{\sqrt{2}\sigma}) & H_2(\frac{z}{\sqrt{2}\sigma}) \end{pmatrix}$$

La matrice hessiana descrive la struttura al secondo ordine delle variazioni locali di intensità attorno ad un punto in una immagine 3D. Valutando la matrice al centro della struttura modellizzata si ottiene:

$$H_{dot}(0, 0, 0) = \begin{pmatrix} -\frac{1}{\sigma^2} & 0 & 0 \\ 0 & -\frac{1}{\sigma^2} & 0 \\ 0 & 0 & -\frac{1}{\sigma^2} \end{pmatrix} \tag{2.9}$$

Da qui si nota come gli autovalori della matrice hessiana siano funzioni della varianza σ. Analogamente si possono ricavare gli autovalori delle matrici hessiane associate alle altre strutture:

$$\begin{aligned} &\text{sfera:} && \lambda_1 = \lambda_2 = \lambda_3 = -1/\sigma^2 \\ &\text{cilindro:} && \lambda_1 = \lambda_2 = -1/\sigma^2 \quad , \quad \lambda_3 = 0 \\ &\text{piano:} && \lambda_1 = -1/\sigma^2 \quad , \quad \lambda_2 = \lambda 3 = 0 \end{aligned} \tag{2.10}$$

dove i λ_i sono gli autovalori ordinati t.c. $|\lambda_1| \geq |\lambda_2| \geq |\lambda_3|$. Ponendo $e_2 = |\lambda_2|/|\lambda_1|$ e $e_3 = |\lambda_3|/|\lambda_1|$, le condizioni precedenti possono essere riscritte:

$$\begin{aligned} &\text{sfera:} && e_2 = 1 \quad , \quad e_3 = 1 \\ &\text{cilindro:} && e_2 = 1 \quad , \quad e_3 = 0 \\ &\text{piano:} && e_2 = 0 \quad , \quad e_3 = 0 \end{aligned} \tag{2.11}$$

Dalle formule (2.10) e (2.11) si nota che la condizione $e_3 = 1$ è verificata solo per la funzione sfera. Inoltre l'autovalore λ_3 è diverso da 0 solo nel caso di struttura sferica. Combinando assieme queste due relazioni si definisce l'output dell'algoritmo *dot-enhancer* come:

$$z_{dot}(\lambda_1, \lambda_2, \lambda_3) = \begin{cases} \frac{|\lambda_3|^2}{|\lambda_1|}, & \text{se } \lambda_1 < 0,\ \lambda_2 < 0,\ \lambda_3 < 0 \\ 0, & \text{altrimenti} \end{cases} \tag{2.12}$$

Ad ogni punto dell'immagine da analizzare è quindi possibile associare un valore legato alla *similitudine* di quel punto con il centro di una struttura sferica (z_{dot}).

2.4.2 Approccio multi-scala

Nel filtro presentato non è stato introdotto nessun parametro di scala. All'interno del volume polmonare i noduli hanno varie dimensioni ed è necessario implementare un approccio che permetta di identificare oggetti in un determinato intervallo di dimensioni.

Il calcolo diretto delle derivate in un'immagine è un problema mal posto e tale da provocare un aumento del rumore già presente nell'immagine. Un metodo per risolvere il problema è convolvere l'immagine con una funzione gaussiana prima del calcolo delle derivate [25, 26]. Inoltre questi due passaggi possono essere combinati in uno solo: la convoluzione fra l'immagine e le derivate di una funzione di *smoothing*[2] gaussiana [26, 16]:

$$\partial_x(f * G(x,\sigma)) = f * \partial_x(G(x,\sigma)) \tag{2.13}$$

dove il simbolo $*$ si riferisce al prodotto di convoluzione fra due segnali:

$$(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau$$

In questo modo viene selezionata una scala di interesse, rendendo possibile un approccio multi-scala.

Un oggetto di profilo gaussiano con parametro di scala σ_0 può essere pensato come avente un diametro di $4\sigma_0$: l'area sottesa in questo intervallo è infatti più del 95% dell'area sottesa su tutto l'asse reale. Quindi se gli oggetti che vogliamo identificare hanno dimensioni aspettate comprese nell'intervallo $[d_{min}, d_{max}]$, si utilizzano fattori di scala nel range $[\sigma_{min}, \sigma_{max}]$ con $\sigma = d/4$. L'intervallo è successivamente campionato in N passi in modo esponenziale [26]:

$$\sigma_i = r^{i-1}\sigma_{min} \quad , \quad i = 1, \ldots, N \tag{2.14}$$

dove

$$r = \left(\frac{d_{max}}{d_{min}}\right)^{1/(N-1)}$$

Le derivate spaziali calcolate secondo (2.13) hanno un'ampiezza che decresce con la scala σ con cui viene effettuato lo smoothing. Per rendere comparabili le risposte del filtro alle varie scale [27] si introduce il concetto di coordinata normalizzata

$$\xi = x/\sigma$$

e dell'operatore adimensionale

$$\partial_\xi = \sigma\partial_x$$

[2]Per una spiegazione sullo smoothing di immagini fare riferimento a [15]

Figura 2.10: Esempi di strutture tridimensionali di profilo gaussiano per differenti valori del parametro di scala T (eq 2.17).

Si nota quindi che ogni operazione di derivata viene soppressa di un fattore σ. L'effetto viene corretto introducendo:

$$z_{norm}(\sigma_i) = \sigma_i^2 \cdot z_{dot}(\sigma_i) \tag{2.15}$$

e l'output del filtro multiscala per ogni voxel è definito dalla z_{norm} più alta fra quelle ottenute alle varie scale:

$$z_{max} = \max_{i \in [1,N]} (z_{norm}(\sigma_i)\,) \tag{2.16}$$

La matrice risultante contenente i valori di z_{max} viene poi analizzata alla ricerca di massimi locali, che identificano le posizioni dei candidati noduli.

Tutti i parametri del filtro devono essere adattati sperimentalmente in base alla dimensione dei noduli da rilevare.

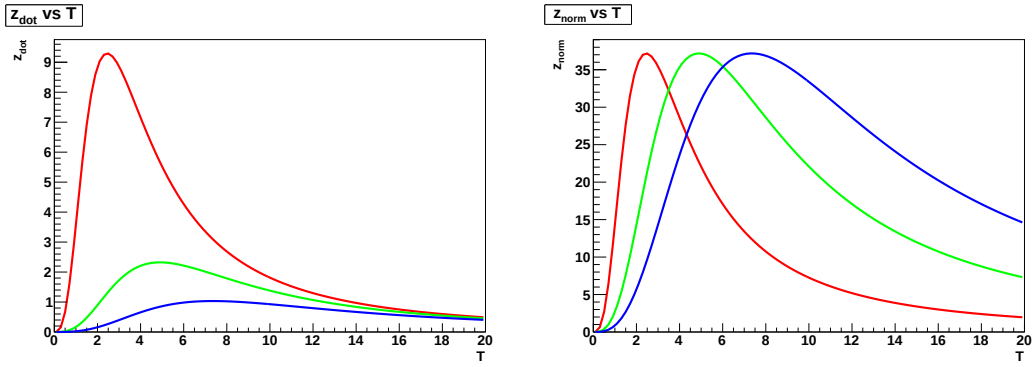
2.4.3 Risposta su oggetti artificiali

È possibile calcolare teoricamente la risposta del filtro su un oggetto tridimensionale artificiale di profilo gaussiano

$$G_T = t_0 \cdot exp\Big(-\frac{|\vec{x}|^2}{2T^2}\Big) \tag{2.17}$$

dove t_0 è una costante positiva e $\vec{x} = (x, y, z)$. In figura 2.10 sono riportati alcuni esempi di questi oggetti per differenti valori del parametro di scala T.

Analiticamente è possibile svolgere prima la convoluzione e poi il calcolo delle derivate seconde richieste dalla matrice hessiana. La funzione di



(a) **(b)**

Figura 2.11: Confronto delle curve di risposta per varie scale del filtro (curva rossa $\sigma = 2$, curva verde $\sigma = 4$, curva blu $\sigma = 6$). Nel riquadro (a) è riportato il profilo di z_{dot} in funzione della dimensione del nodulo simulato, nel riquadro (b) invece è riportato il profilo della funzione z_{norm}.

convoluzione è una gaussiana normalizzata:

$$G_\sigma = \frac{1}{(\sigma\sqrt{2\pi})^3} \cdot exp\Big(-\frac{|\vec{x}|^2}{2\sigma^2}\Big) \tag{2.18}$$

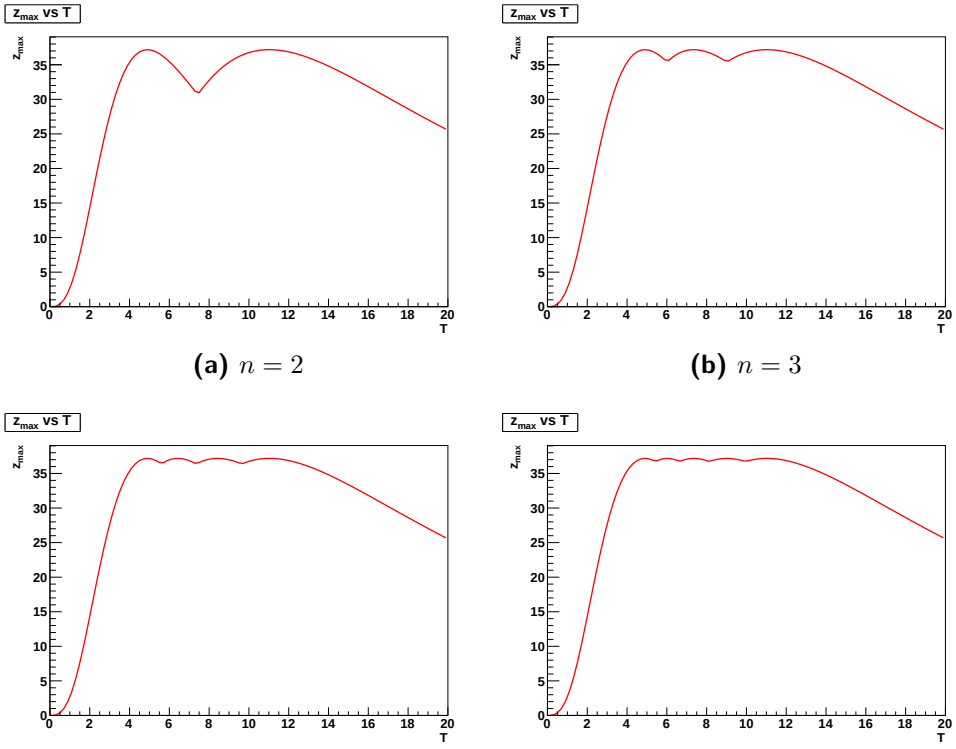
Il risultato della convoluzione di (2.18) con (2.17) è sempre una gaussiana il cui parametro di scala è $\sqrt{\sigma^2 + T^2}$. Svolgendo i calcoli (ad esempio sfruttando la trasformata di Fourier), si ottiene:

$$G_T * G_\sigma = t_0\Big(\frac{T}{\sqrt{\sigma^2 + T^2}}\Big)^3 exp\Big(-\frac{|\vec{x}|}{2(\sigma^2 + T^2)}\Big) \tag{2.19}$$

Utilizzando il risultato (2.9), con il nuovo parametro di scala $\sqrt{\sigma^2 + T^2}$ e la definizione di z_{dot} (2.12) si ottiene:

$$z_{dot}(\sigma, T) = t_0 \frac{T^3}{(\sigma^2 + T^2)^{\frac{5}{2}}} \tag{2.20}$$

In figura 2.11 è riportata la curva di risposta sia per la funzione z_{dot} e z_{norm} per vari parametri del filtro, al variare della dimensione T del nodulo simulato. Si può notare come il valore del massimo della funzione z_{dot} decresca con l'aumentare della scala σ del filtro ($\propto 1/\sigma^2$). La funzione z_{norm} invece non presenta questo andamento: i valori massimi di risposta del filtro sono



(a) $n = 2$ **(b)** $n = 3$

(c) $n = 4$ **(d)** $n = 5$

Figura 2.12: Andamento della risposta del filtro multiscala z_{max} al variare del numero di suddivisioni dell'intervallo (n). Gli altri parametri del filtro sono $\sigma_{min} = 4$, $\sigma_{max} = 9$, $t_0 = 200$.

comparabili fra loro alle varie scale, permettendo quindi un'analisi multiscala. Il massimo della funzione $z_{dot}(T)$ (2.20) si ottiene per:

$$T = \sigma\sqrt{\frac{3}{2}}$$

e non per $T = \sigma$. Questo è importante perchè, se vogliamo utilizzare correttamente questo filtro, è necessario ricalibrare le scale.

Anche per l'algoritmo multiscala è possibile fare un grafico della funzione z_{dot} al variare del parametro T della struttura di test. In figura 2.12 è riportato l'andamento teorico della funzione z_{max} (2.16) al variare del numero di passi dell'algoritmo ($n = 2, \ldots, 5$), lasciando invariato l'intervallo di analisi. Possiamo notare come, all'aumentare del numero di passi intermedi, la risposta del filtro sia sempre più stabile all'interno dell'intervallo di interesse. Utilizzando la formula (2.16) in si ottiene il grafico riportato in figura 2.12.

Capitolo 3

Graphical Processing Unit e architettura CUDA

3.1 Cosa sono le GPU

Negli ultimi anni l'industria dei semiconduttori si è orientata verso due principali paradigmi di sviluppo di processori: le *multi-core* CPU (Central Processing Unit) e le *many-core* GPU (Graphical Processing Unit) [28]. Queste ultime, progettate con l'obiettivo di poter effettuare un maggior numero di operazioni in virgola mobile per ogni fotogramma da visualizzare, hanno prodotto un notevole aumento di prestazioni (figura 3.1).

Il motivo per cui esiste una così grande discrepanza nella capacità di eseguire operazioni floating point fra CPU e GPU è dovuta alla loro struttura interna [30]. Le CPU sono progettate per ottimizzare le prestazioni dei programmi sequenziali: utilizzano una logica di controllo sofisticata per permettere alle istruzioni di un singolo *thread*[1] di essere eseguite in modo parallelo o anche fuori dal loro ordine sequenziale, mantenendo l'apparenza di una esecuzione sequenziale, in modo da eseguire lo stesso numero di istruzioni in minor tempo. Una gran parte del processore è dedicata alla memoria *cache*, utilizzata per diminuire la "*latenza*"[2] del sistema al fine di rendere disponibili i dati per successive richieste ravvicinate nel tempo. Le GPU invece sono progettate per computazioni intense e altamente parallele (come il *rendering* di immagini) e, quindi, molti più transistor sul chip sono dedicati all'elaborazione

[1] in informatica un *thread* è una suddivisione di un processo in due o più filoni, che vengono eseguiti concorrentemente dal processore. Per *processo* si intende invece un'istanza di un programma in esecuzione in modo sequenziale.

[2] con il termine "*latenza*" di un sistema informatico si intende l'intervallo di tempo che intercorre tra la richiesta di un dato alla memoria e la sua disponibilità per essere elaborato.

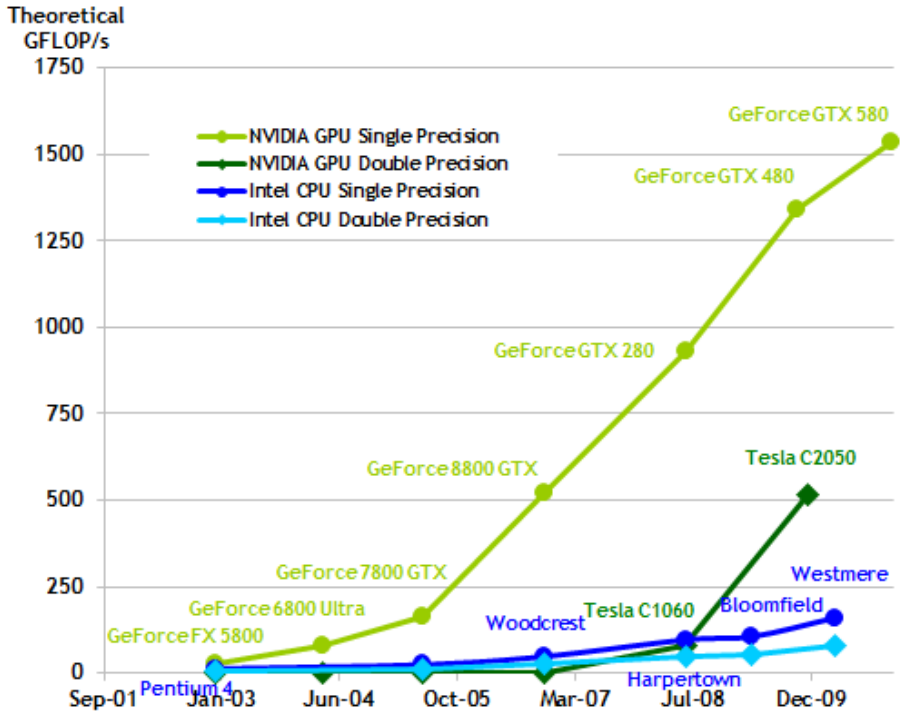


Figura 3.1: CPU vs. GPU: evoluzione della potenza di calcolo teorica, andamento degli ultimi anni [29].

dei dati, piuttosto che alla loro memorizzazione per un frequente utilizzo (memoria cache) e per il controllo del flusso di esecuzione (figura 3.2).

In particolare, le GPU sono adatte a risolvere problemi che possono essere espressi come calcoli paralleli sui dati: il dato viene partizionato e lo stesso programma viene eseguito in parallelo su tutti gli elementi. Con questo paradigma non è più necessaria una logica complessa per la gestione del flusso del programma né una grande dimensione della memoria cache, poiché la latenza degli accessi in memoria può essere ammortizzata grazie all'alto numero di operazioni eseguite.

Le prime generazioni di questi dispositivi erano progettate per svolgere solo alcuni tipi di operazioni, con l'obiettivo di elaborare dati per la loro visualizzazione su schermo. L'interesse della comunità scientifica verso questi dispositivi è iniziata quando è stata introdotta la possibilità di configurare e programmare questi dispositivi in modo da poter eseguire programmi generici (General Purpose programming using Graphic Processing Unit - GPGPU).

Dal 2007 questo è cambiato con l'introduzione dell'architettura CUDA (Compute Unified Device Architecture) da parte di NVIDIA [31]. Questa architettura per il calcolo parallelo ha l'obiettivo di semplificare lo sviluppo di

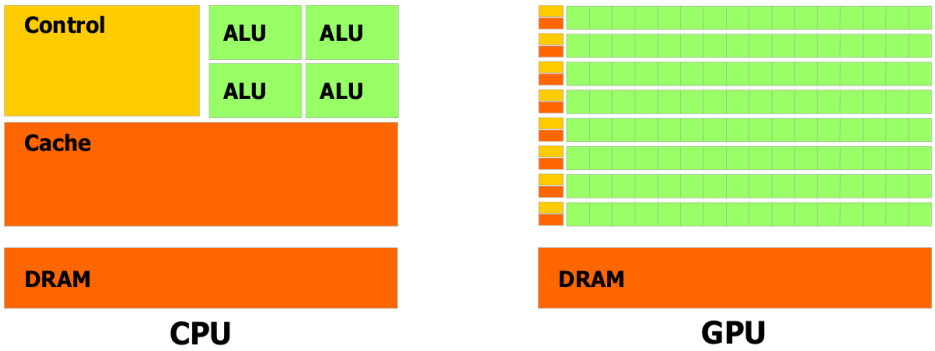


Figura 3.2: Confronto fra l'area (e quindi il numero di transistor) dedicata all'elaborazione dei dati, alla gestione del flusso del programma e alla cache [29].

applicazioni su GPU: un chip dedicato sulla GPU si occupa di gestire il flusso del programma e permette l'esecuzione di migliaia di threads in parallelo. Inoltre, per programmare questi dispositivi sono state realizzate semplici estensioni al linguaggio di programmazione C/C++. Nel paragrafo 3.2 è descritta la struttura di una GPU e l'architettura CUDA. Nel capitolo seguente (paragrafo 4.2) è descritta l'implementazione dell'algoritmo *dot-enhancer* su GPU.

3.2 CUDA in dettaglio

Al contrario della programmazione generica su CPU, scrivere programmi per questi dispositivi richiede una conoscenza più approfondita dell'architettura hardware, per sfruttare efficacemente le potenzialità delle GPU.

Come già accennato, per trarre vantaggio da questo tipo di architettura *multi-thread* (MT) parallela è fondamentale che il problema presenti un forte parallelismo sui dati. Ne è un esempio il prodotto interno fra matrici, in cui ogni valore della matrice risultante è indipendente dagli altri, e quindi può essere calcolato in parallelo contemporaneamente agli altri.

3.2.1 Modello hardware

L'elemento fondamentale dell'architettura HW di CUDA sono gli *Streaming Multiprocessor* (SM) [32, 30]; questi sono collocati all'interno di un unico chip, come schematizzato in figura 3.3. L'architettura corrente (denominata

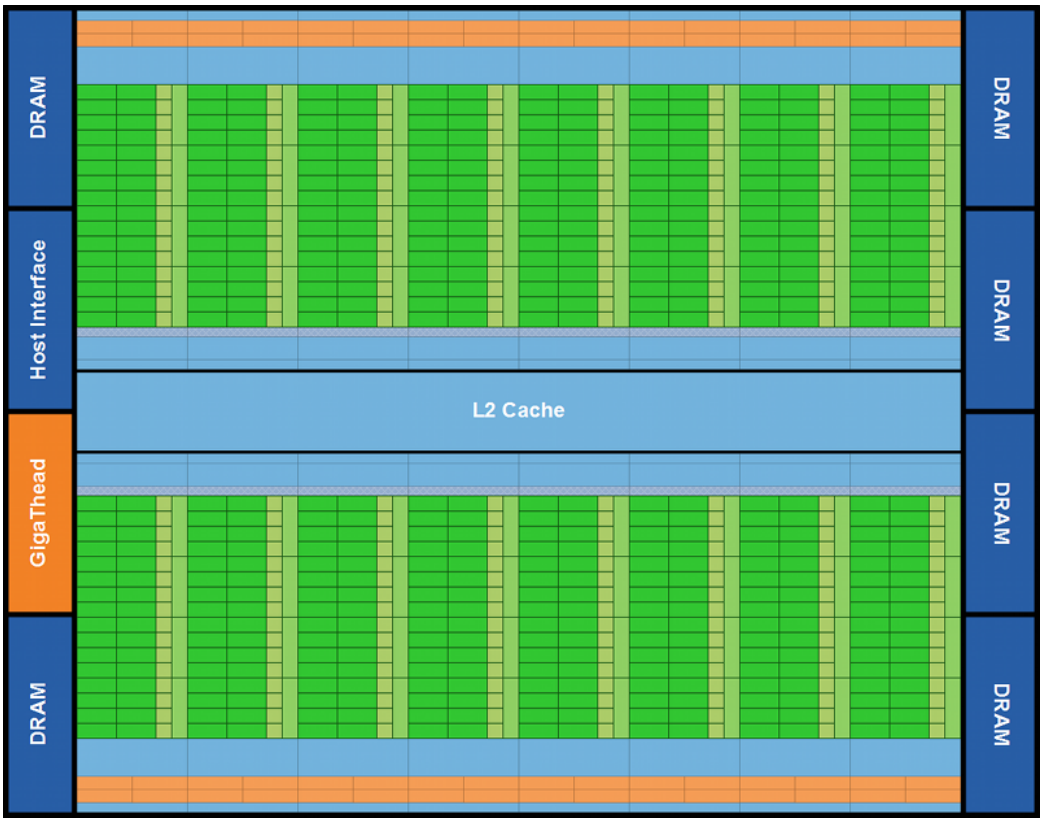


Figura 3.3: Architettura NVIDIA Fermi: 16 Streaming Multiprocessor sono posizionati attorno a una cache L2 comune. Ogni SM è una striscia verticale rettangolare che contiene una parte color arancio (*scheduler*), una parte verde (le unità di esecuzione) e le parti blu chiaro (registri e cache L1) [32].

Fermi) è costituita da 16 di questi processori ciascuno dei quali contiene due gruppi di 16 *Scalar Processor* (SP) o *CUDA Core* (figura 3.4).

Ognuno di questi SP contiene una *integer arithmetic logic unit* (ALU) e una *floating point aritmetic unit* (FPU). Inoltre ogni SM dispone di unità per il calcolo di funzioni trascendenti, chiamate Special Function Unit (SFU). Ogni core può eseguire un singolo thread, ma tutti i core dello stesso gruppo eseguono la stessa istruzione allo stesso tempo: questo paradigma è chiamato Single Instruction Multiple Data (SIMD)[3].

Esistono vari tipi di memorie nell'architettura HW di CUDA che compongono una gerarchia simile a quella dei normali computer. Una memoria *global*

[3]in realtà NVidia chiama questo paradigma SIMT, a causa della differente gestione del flusso del programma per ciascun thread a seguito di istruzioni condizionali.

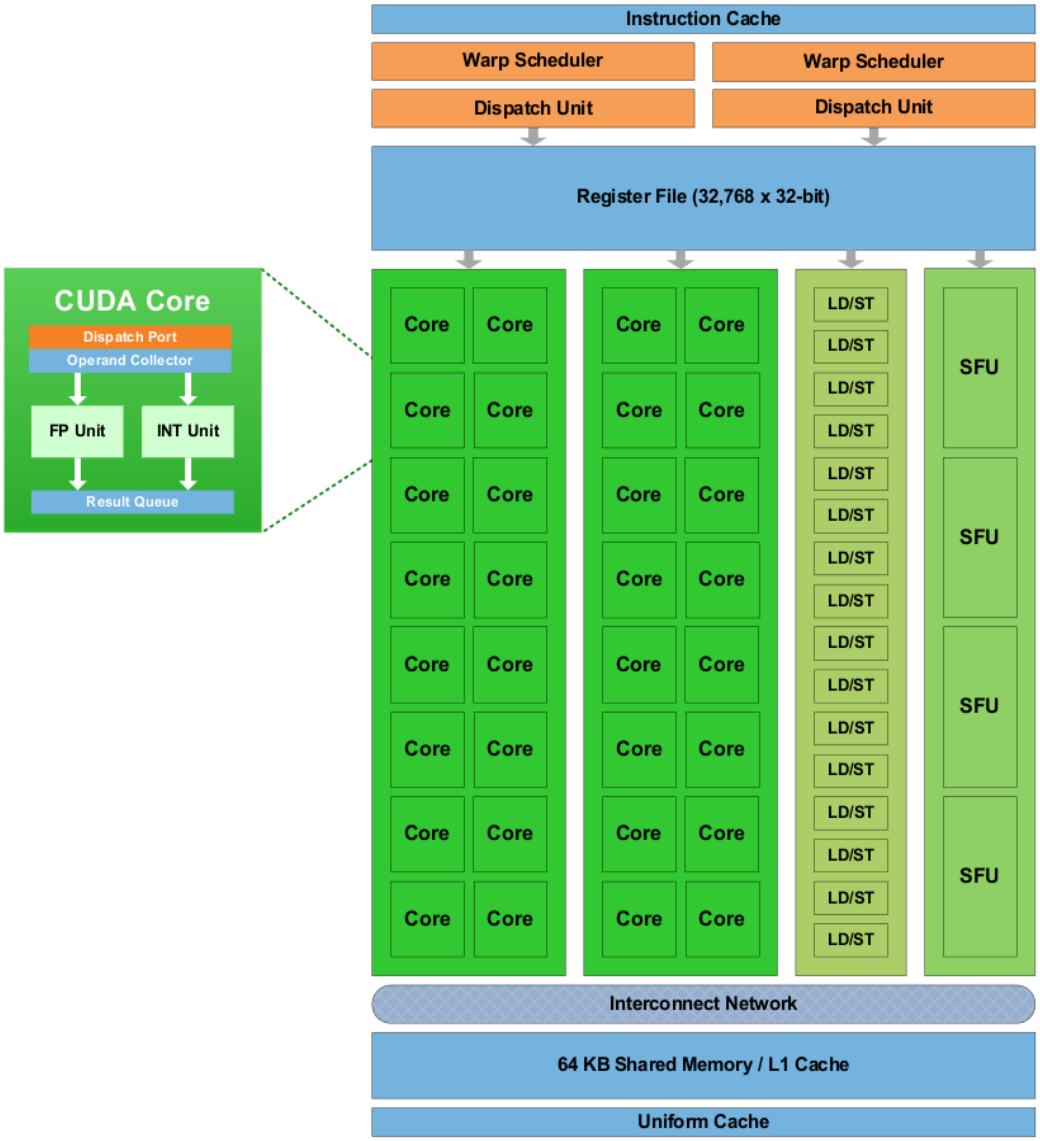


Figura 3.4: Architettura di uno Streaming Multiprocessor per l'architettura Fermi: sono visibili i 32 CUDA core (il riquadro a lato ne evidenzia un particolare), le unità per il calcolo delle funzioni trascendenti (FSU) e le varie memorie *on-chip* (registri e *shared memory*), oltre ad altre logiche di controllo [32].

accessibile da tutti i SM della GPU, non molto veloce rispetto agli altri tipi di memoria presenti sulla GPU, ma solitamente molto grande (dell'ordine del gigabyte). In aggiunta a questa memoria, ciascun SM dispone di varie tipologie di memoria, condivisa da tutti gli SP del core:

- Un insieme di *registri* a 32 bit, il cui accesso non richiede nessun ciclo di clock aggiuntivo.
- Una memoria *shared* di 64 KB, a bassa latenza e grande banda passante (paragonabile ai registri), condivisa da tutti i threads in esecuzione sullo stesso SM.
- Una memoria *constant*, di sola lettura, inizializzata dal software all'inizio dell'esecuzione ed utilizzata per il rapido accesso a dati costanti durante il flusso del programma.

Comprendere la struttura gerarchica delle memorie in questa architettura è fondamentale per l'efficienza del programma. È importante infatti riuscire a minimizzare la necessità di accedere alla memoria globale, gestendo la struttura del calcolo in modo da prediligere, per la parte computazionalmente intensiva, l'accesso alla memoria locale di ogni SM (shared e constant).

Il numero di thread che viene eseguito contemporaneamente su un singolo SP è chiamato *warp*: nell'architettura attuale è composto da 32 thread. Quando un thread esegue un'operazione sulla memoria, quell'istruzione per essere eseguita occuperà un tempo lungo rispetto alle operazioni sui registri[4], a causa della grande latenza della memoria globale. Le architetture come i processori multicore aggiungono molti livelli di cache per ammortizzare queste latenze. Le GPU invece sfruttano il gran numero di thread in esecuzione per ammortizzare queste latenze: vengono tenuti attivi più warp sullo stesso SP, in modo che quando un warp è in stallo in attesa di dati dalla memoria globale, il successivo viene subito messo in esecuzione al suo posto. Questa procedura è molto efficiente perché è gestita a livello hardware (*dispatch unit* in figura 3.4).

Un ultimo concetto collegato alla gestione della memoria riguarda l'accesso alla memoria globale. Per ovviare alla lentezza degli accessi a questo tipo di memoria (fondamentale per la memorizzazione dei dati da analizzare) ad ogni richiesta di accesso, l'harware permette di leggere o scrivere un intero blocco di dati contigui. Se quindi tutti i threads di un warp accedono a dati contigui questi accessi possono essere effettuati contemporaneamente (*coalescence* nel gergo di CUDA), contribuendo ad abbattere il costo dell'accesso alla memoria globale.

[4]anche alcune centinaia di volte superiore al tempo di accesso ai registri

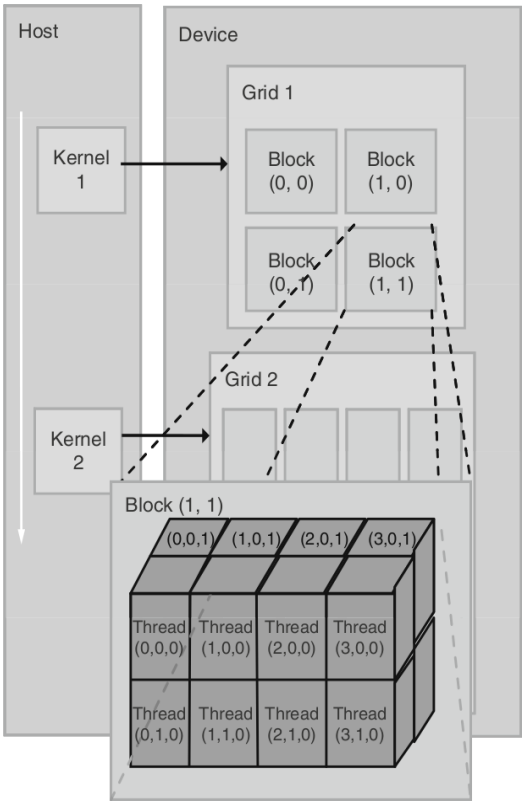


Figura 3.5: Gerarchia dei threads nell'architettura CUDA.

3.2.2 Modello di programmazione

Un programma CUDA si compone di più parti che possono essere eseguite sia sulla CPU (*host*) o sulla GPU (chiamata *device*). L'architettura CUDA si occupa di rendere visibile la GPU come un coprocessore parallelo alla CPU. Generalmente la parte difficilmente parallelizzabile del programma viene eseguita sulla CPU, mentre il resto sulla GPU. La parte di programma eseguita su CPU si occupa inoltre di trasferire i dati dalla memoria RAM dell'host sulla memoria globale del device: non è possibile per i programmi che vengono eseguiti sul device accedere direttamente alla memoria dell'*host*.

Una funzione eseguita sul device viene chiamata *kernel*. I kernel sono organizzati in una gerarchia di threads (figura 3.5). I threads sono raggruppati in blocchi (*threads blocks*) e i blocchi riuniti in una griglia (*grid*):

block un blocco di thread è un insieme di threads eseguiti in modo concorrente, che possono sincronizzarsi fra di loro e lavorare su dati condivisi. A ogni thread del blocco è assegnato dinamicamente un indice tridimen-

sionale unico, utilizzato da ciascun thread per accedere ai dati. Ogni blocco viene assegnato a un singolo SM.

grid una griglia di blocchi è un insieme utilizzato per raggruppare i blocchi. Poiché in generale è possibile che griglie differenti siano allocate su SM diversi, i thread appartenenti alla stessa griglia non possono utilizzare la memoria shared per scambiarsi dati, ma possono accedere solo alla memoria globale. Ogni blocco ha il suo indice bidimensionale.

Ogni thread dello stesso kernel esegue le stesse operazioni ma su dati differenti: l'accesso a questi dati avviene utilizzando gli indici propri associati a ciascun thread. Questa pluridimensionalità della gerarchia dei thread può aiutare nell'implementazione degli algoritmi creando una corrispondenza fra i singoli thread e i dati.

Per sfruttare completamente le potenzialità di una GPU è necessario quindi che siano verificate le seguenti condizioni:

- un alto livello di intensità aritmetica, ovvero più operazioni numeriche che accessi in memoria per ogni kernel da eseguire.
- un alto grado di parallelismo, in modo che tutti i SP siano completamente utilizzati.
- un accesso alla memoria prefissato, per organizzare i threads in modo ottimale per l'accesso alla memoria globale.

È inoltre fondamentale cercare di ridurre il traffico dati fra CPU e GPU, poichè spesso è uno dei colli di bottiglia nell'implementazione di algoritmi su GPU.

Capitolo 4

Ottimizzazione e implementazione su GPU dell'algoritmo *dot-enhancer*

In questo capitolo è descritta l'implementazione dell'algoritmo di identificazione dei candidati noduli *dot-enhancer* [16] descritto nel paragrafo 2.4. Sono presentate le varie ottimizzazioni realizzate per la sua implementazione su CPU e l'implementazione basata su GPU. I risultati ottenuti saranno descritti nel capitolo 5. Questo algoritmo, grazie al suo parallelismo sui dati si presta in maniera diretta ad una implementazione su GPU. In questo modo è stato possibile sperimentare la loro efficacia nella riduzione dei tempi di esecuzione del CAD.

4.1 Ottimizzazioni dell'algoritmo *dot-enhancer*

L'algoritmo dot-enhancer descritto al paragrafo 2.4 è riassunto nello schema 1.

Questo algoritmo è computazionalmente molto intensivo, a causa del grande numero di voxel da analizzare: tipicamente una scansione TAC è composta da 10^8 voxel. Sono state effettuate quindi alcune ottimizzazioni nei punti più critici (linee 6 e 4).

Per il calcolo degli elementi della matrice hessiana (linea 4), l'algoritmo utilizzato per il filtraggio è di tipo ricorsivo [33], la cui complessità (alle scale di interesse) è minore rispetto a un filtraggio di tipo FIR (Finite Impulse Response) o all'utilizzo di operazioni di trasformata di Fourier.

Schema 1 Algoritmo *dot-enhancer*

Input: Immagine originale $I(i,j,k)$, numero di passi N, intervallo $[d_{min}, d_{max}]$
Output: $Z_{max}(i,j,k)$ matrice di valori z_{max}

1: **foreach** $i \in \{1,\ldots,N\}$ **do** $\sigma_i \leftarrow \sigma_{min} \cdot (\frac{\sigma_{max}}{\sigma_{min}})^{\frac{(i-1)}{(N-1)}}$
2: **foreach** (i,j,k) **do** $Z_{max}(i,j,k) \leftarrow 0$ ▷ inizializzazioni
3: **for** $i \leftarrow 1, N$ **do**
4: calcolo componenti matrice hessiana $(H_{xx}, H_{xy}, H_{xz}, H_{yy}, H_{yz}, H_{zz})$
5: **for all** (i,j,k) **do**
6: calcolo autovalori $(\lambda_1, \lambda_2, \lambda_3)$
7: $z_{norm} \leftarrow \sigma_i^2 \cdot |\lambda_3|^2 / |\lambda_1|$
8: **if** $Z_{max} < z_{norm}$ **then**
9: $Z_{max} \leftarrow z_{norm}$
10: **end if**
11: **end for**
12: **end for** ▷ fine algoritmo

Il costo computazionale del calcolo degli autovalori (linea 6) è stato ottimizzato sfruttando la soluzione analitica del polinomio caratteristico e il criterio di Liénard-Chipart [34]. Gli autovalori cercati sono le radici del polinomio caratteristico della matrice hessiana, che è della forma:

$$\lambda^n + a_1\lambda^{n-1} + \cdots + a_n = 0 \tag{4.1}$$

con $n = 3$. Generalmente per il calcolo degli autovalori si utilizzano metodi iterativi [35], che risultano essere eccessivamente onerosi per il caso $n = 3$. È infatti possibile ricavare le radici con la formula diretta dell'equazione cubica, la cui espressione si può trovare in [36]. Gli autovalori della matrice hessiana (essendo simmetrica) sono reali.

Come è stato discusso in [37], l'implementazione della formula diretta può portare a instabilità numeriche. Tuttavia, dato che per questo tipo di applicazione un errore dell'ordine del percento è tollerabile, questa formula conduce a risultati soddisfacenti. Infatti, per quanto emerso dagli esperimenti eseguiti in [37] e in questo lavoro di tesi, la formula diretta, nel calcolo degli autovalori, ha un'accuratezza minima dell'ordine di 10^{-4}.

Poichè la funzione z_{norm} è diversa da 0 solo se gli autovalori sono negativi (equazione 2.12 e 2.15), si utilizzata un criterio di stabilità per controllare questa condizione prima di effettuare il calcolo. Il criterio utilizzato è il teorema di Liénard-Chipart [34]. Indicando con Δ_i il determinante di Hurwitz

associato all'equazione (4.1):

$$\Delta_i = \begin{vmatrix} a_1 & a_3 & . & . & a_{2i-1} \\ 1 & a_2 & . & . & a_{2i-2} \\ 0 & a_1 & . & . & a_{2i-3} \\ . & . & . & . & . \\ 0 & 0 & . & . & a_i \end{vmatrix}$$

Teorema 1. *(criterio di Liénard-Chipart) Affinché le parti reali di tutte le radici dell'equazione* (4.1) *siano negative è necessario e sufficiente che sussistano le seguenti disequazioni*[1]*:*

$$a_n > 0, \quad a_1 > 0, \quad a_3 > 0, \quad a_5 > 0, \ldots \tag{4.2}$$

$$\Delta_{n-1} > 0, \quad \Delta_{n-3}, \quad \Delta_{n-5}, \ldots, \begin{cases} \Delta_3 > 0 & n \textit{ pari} \\ \Delta_2 > 0 & n \textit{ dispari} \end{cases} \tag{4.3}$$

Nel caso $n = 3$ le condizioni diventano:

$$\begin{aligned} a_3 &> 0 \\ a_1 &> 0 \\ a_1 a_2 - a3 &> 0 \end{aligned} \tag{4.4}$$

dove i coefficienti a_i sono dipendenti dai 6 valori indipendenti della matrice hessiana H_{ij}. Se le condizioni (4.4) non sono verificate, gli autovalori non vengono calcolati.

Con questi accorgimenti è stato possibile ridurre sensibilmente i tempi di computazione dell'algoritmo implementato su CPU. I risultati numerici sono presentati nel capitolo 5.

4.2 Implementazione dell'algoritmo *dot-enhancer* su GPU

Come primo esperimento di traduzione degli algoritmi del CAD su architettura GPU è stato scelto l'algoritmo dot-enhancer. Questo algoritmo ha una naturale struttura parallela sui dati. Sia il calcolo della matrice hessiana che quello degli autovalori possono infatti essere implementati in modo che un insieme di kernel elabori le informazioni di ciascun voxel dell'immagine in modo parallelo. Inoltre i filtri basati sull'analisi degli autovalori della matrice hessiana trovano larga applicazione nell'analisi di immagini mediche, come

[1] sequenze alternative a (4.2) e (4.3) sono possibili

ad esempio per l'identificazione di strutture vascolari. Ottenere quindi un vantaggio computazionale in questo tipo di algoritmi permette di avere uno strumento utile anche per successive applicazioni.

4.2.1 Concetti fondamentali

L'algoritmo presentato nella sezione 2.4 è basato sul calcolo della matrice hessiana e dei suoi autovalori.

Per ogni voxel è necessario calcolare la matrice hessiana associata all'immagine originale convolvendo l'immagine con le opportune derivate della Gaussiana (paragrafo 2.4.2). In questo modo si ottengono 6 matrici tridimensionali: H_{ij} con $i, j \in x, y, z$ e $H_{ij} = H_{ji}$ dove i e j indicano le direzioni lungo le quali è stata effettuata l'operazione di derivazione (i.e. H_{xx} è il risultato della convoluzione con la derivata seconda della funzione gaussiana). Da queste matrici si ricava la matrice hessiana associata a ciascun voxel, utilizzando i valori dei rispettivi voxel nelle matrici filtrate.

La versione discreta dell'operazione di convoluzione su una immagine $I(i, j, k)$ è solitamente [15] implementata come:

$$\tilde{I}(i,j,k) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \sum_{v=-c}^{c} I(s,t,v)h(i+s, j+t, k+v) \tag{4.5}$$

dove h è la risposta all'impulso del sistema (o maschera di convoluzione).

Questo algoritmo è stato scelto perchè permette una semplice ed efficace implementazione parallela. Il numero di operazioni (somme e moltiplicazioni) necessarie per implementare la convoluzione, utilizzando maschere cubiche di lato N, è proporzionale a N^3.

È possibile ridurre questo fattore a $3N$, sfruttando la proprietà della *separabilità* della Gaussiana e delle sue derivate.

Dalla definizione di convoluzione di un segnale $f(x, y, z)$ con una gaussiana $G_\sigma(x, y, z) = e^{-(x^2+y^2+z^2)/2\sigma}$, si ha:

$$\begin{aligned} \tilde{f}(x,y,z) &= (f * G)(x,y,z) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau_x,\tau_y,\tau_z)G_\sigma(x-\tau_x, y-\tau_y, z-\tau_z)d\tau_x d\tau_y d\tau_z \end{aligned} \tag{4.6}$$

utilizzando:

$$G(x,y,z) = e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} \tag{4.7}$$

si ha:

$$\begin{aligned} \tilde{f} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau_x,\tau_y,\tau_z)(e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}})d\tau_x d\tau_y d\tau_z \\ &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} d\tau_x \int_{-\infty}^{\infty} e^{-\frac{y^2}{2\sigma^2}} d\tau_y \int_{-\infty}^{\infty} f(\tau_x,\tau_y,\tau_z) e^{-\frac{z^2}{2\sigma^2}} d\tau_z \end{aligned}$$

quindi:

$$\tilde{f} = G_\sigma(x) * G_\sigma(y) * (f * G_\sigma(z)) \tag{4.8}$$
$$= ((f * G_\sigma(x)1,) * G_\sigma(y)) * G_\sigma(z) \tag{4.9}$$

Ogni convoluzione con una gaussiana tridimensionale è quindi decomponibile in tre convoluzioni (una per ogni coordinata o direzione) con una funzione gaussiana unidimensionale. La proprietà vale anche per le convoluzioni con le derivate della gaussiana. In questo modo ognuna delle 6 matrici necessarie si ottiene da una sequenza di tre convoluzioni con un segnale unidimensionale. Dall'equazione (4.5) si ha:

$$\begin{aligned} \tilde{I}_x(i,j,k) &= \sum_{t=-a}^{a} I(t,j,k)h(i+t) \\ \tilde{I}_y(i,j,k) &= \sum_{t=-a}^{a} I(i,t,k)h(j+t) \\ \tilde{I}_z(i,j,k) &= \sum_{t=-a}^{a} I(i,j,t)h(k+t) \end{aligned} \tag{4.10}$$

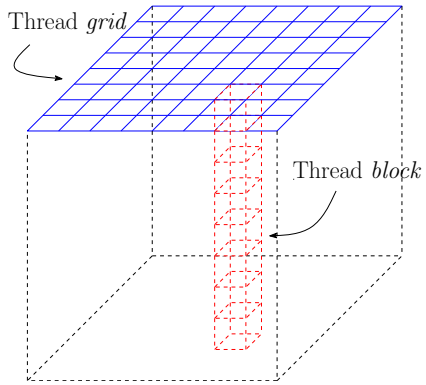
La risposta all'impulso per il sistema è ottenuta dal campionamento della funzione gaussiana (e delle sue derivate) in punti corrispondenti al centro di ogni voxel. La dimensione della maschera è determinata in funzione della scala σ: la gaussiana è troncata a 0 per valori di $x > 4\sigma$. L'errore commesso rispetto all'area della gaussiana è inferiore allo $0.1‰$, sufficiente per il corretto funzionamento di questo filtro.

4.2.2 Dettagli implementativi

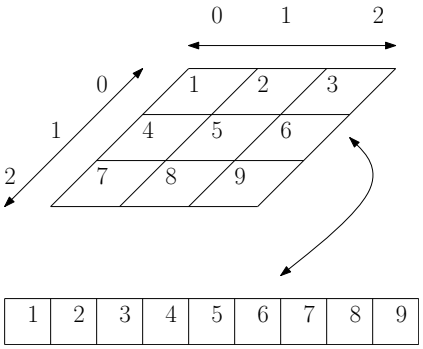
L'elemento base per ottenere le matrici H_{ij} è quindi l'operazione di convoluzione. Dalle equazioni (4.5) e (4.10) si nota come sia possibile eseguire la convoluzione in modo completamente parallelo: è possibile associare a ciascun voxel dell'immagine un thread che esegua la convoluzione per quell'indice dell'immagine.

L'organizzazione dei threads utilizzata è riportata in figura 4.1a. Ogni blocco di thread lavora in modo cooperativo per eseguire la convoluzione di un'intera riga della matrice originaria lungo la direzione richiesta. La griglia dei thread è strutturata in modo che ciascun voxel dell'immagine abbia un thread associato. Ogni blocco di thread è strutturato per sfruttare la memoria shared e minimizzare le richieste di dati alla memoria globale.

La direzione lungo cui viene effettuata la convoluzione ha un effetto sulle prestazioni dell'algoritmo. Il motivo di questo fenomeno è dato dal fatto che



(a) Organizzazione dei *threads* utilizzata nell'implementazione dell'algoritmo *dot-enhancer*.



(b) Disposizione lineare dei dati in memoria.

Figura 4.1

la matrice tridimensionale viene memorizzata come una sequenza lineare di valori (figura 4.1b).

Lungo la direzione x i dati sono quindi memorizzati in modo sequenziale. Invece, cercando di accedere alla memoria lungo la direzione y o z, non si incontrano aree di memoria contigue. A causa di ciò solo lungo la direzione x gli accessi alla memoria globale possono essere raggruppati (*coalesced*) dalla GPU, risultando in una riduzione del tempo necessario all'analisi di tutta la riga.

Utilizzando la proprietà commutativa dell'operazione di convoluzione è possibile ridurre il degrado delle performance dovuto agli accessi *non-cohalesced*. Indicando con H_{ij} il risultato della convoluzione dell'immagine originale I e con G_i^n la derivata n-esima della Gaussiana lungo la direzione $i \in \{x, y, x\}$ si ha:

$$\begin{aligned} H_{xx} &= I * G_x^{''} * G_y * G_z & H_{xy} &= I * G_x^{'} * G_y^{'} * G_z \\ H_{xz} &= I * G_x^{'} * G_y * G_z^{'} & H_{yy} &= I * G_x * G_y^{''} * G_z \\ H_{yz} &= I * G_x * G_y^{'} * G_z^{'} & H_{zz} &= I * G_x * G_y * G_z^{''} \end{aligned} \tag{4.11}$$

e utilizzando $A = I * G_z$ e $B = I * G_z^{'}$, è possibile riscrivere le relazioni precedenti come:

$$\begin{aligned} H_{xx} &= A * G_y * G_x^{''} & H_{xy} &= A * G_y^{'} * G_x^{'} \\ H_{xz} &= B * G_y * G_x^{'} & H_{yy} &= A * G_y^{''} * G_x \\ H_{yz} &= B * G_y^{'} * G_x & H_{zz} &= I * G_x * G_y * G_z^{''} \end{aligned} \tag{4.12}$$

Tabella 4.1: Durata dei *kernel* convolutivi: sono riportati i tempi di esecuzione per una singola convoluzione lungo la direzione riportata (x,y o z) di una matrice di dimensione $256 \times 256 \times 256$.

direzione di convoluzione	tempo (μs)
x	5709
y	8107
z	14297

dove il numero delle convoluzioni da eseguire è ridotto da 18 a 15. Indicando con T_x il tempo necessario a effettuare una convoluzione lungo la direzione x, $T_y = \alpha T_x$ lungo y e $T_z = \beta T_x$ lungo z, si può esprimere il tempo necessario ad eseguire la sequenza di operazioni in (4.11) come $T_{sub-optimal} = 6T_x(1+\alpha+\beta)$, e per la sequenza (4.12) come $T_{optimal} = 6T_x(1+\alpha+\beta/2)$, da cui il guadagno G:

$$G = \frac{T_{sub-optimal} - T_{optimal}}{T_{sub-optimal}} = \frac{\frac{\beta}{2}}{1+\alpha+\beta} \tag{4.13}$$

Dai dati riportati in tabella 4.1 si ricavano i seguenti valori,

$$\alpha = 1.5, \quad \beta = 2.5$$

e sostituendo in (4.13) si ottiene un guadagno del 25%.

La seconda parte dell'algoritmo dot-enhancer riguarda il calcolo degli autovalori della matrice hessiana. Per ogni voxel la matrice hessiana è ottenuta dai corrispondenti valori presenti nelle 6 matrici H_{ij} precedentemente calcolate e l'accesso ai dati è eseguito in modo ottimale, come descritto precedentemente. Anche nell'implementazione su GPU è stata utilizzata la formula diretta per la risoluzione del polinomio caratteristico [36]. Non è stato però utilizzato il criterio di Liénard-Chipart per discriminare il segno degli autovalori (paragrafo 4.1): gli autovalori sono calcolati sempre per ciascun voxel. Nell'architettura CUDA è infatti consigliato che tutti i threads seguano la stessa catena di istruzioni all'interno del medesimo warp. In caso contrario è possibile che si perda il parallelismo e l'esecuzione venga serializzata con un conseguente degrado delle prestazioni [38].

La memoria globale disponibile sulle GPU è limitata a pochi GB, non sufficienti per contenere tutte le matrici intermedie necessarie all'algoritmo. È stato quindi previsto un meccanismo di suddivisione della matrice di input in più sottomatrici di dimensioni ridotte su cui viene eseguito l'algoritmo

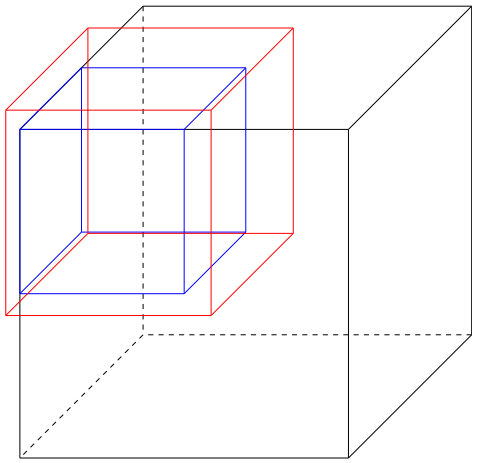


Figura 4.2: Separazione della matrice di input (nero): la sottoimmagine generata per l'analisi è disegnata in rosso, i risultati finali (in blu) sono estratti da quest'ultima.

dot-enhancer. Queste sottomatrici sono fra loro sovrapponibili per evitare che l'analisi presenti artefatti dovuti agli effetti di bordo nelle zone di separazione.

I risultati numerici sono presentati nel prossimo capitolo.

Capitolo 5

Risultati

L'obiettivo cercato implementando l'algoritmo dot-enhancer su GPU è quello di ridurre sensibilmente i tempi di calcolo, mantenendo inalterate le prestazioni nell'identificazione dei candidati noduli interni. In questo capitolo sono presentati i risultati ottenuti in questa direzione.

Inizialmente, nel paragrafo 5.1, sono presentati i guadagni in termini di tempo di esecuzione. Per valutare questi tempi sono state utilizzate immagini reali, poiché sono rappresentative dei dati clinici che devono essere analizzati dagli algoritmi. Il database utilizzato per questo scopo è il database LIDC descritto al capitolo 1, contenente 138 scansioni TAC.

Successivamente, nel paragrafo 5.2, sono riportati i test, effettuati su immagini sintetiche, eseguiti per analizzare la correttezza numerica delle due implementazioni dell'algoritmo

Infine, nel paragrafo 5.3, è descritta la procedura di allenamento del sistema CAD_I, effettuata utilizzando le liste dei candidati noduli provenienti da entrambe le implementazioni realizzate del modulo di ricerca dei noduli interni (RH_I^{CPU} e RH_I^{GPU}). In questo modo è stato possibile confrontarne le prestazioni su dati reali. Il sistema CAD_I, il cui modulo di identificazione dei candidati noduli interni è implementato su GPU, è identificato dalla sigla CAD_I^{GPU} e analogamente è definito il sistema CAD_I^{CPU}.

Per ottenere i risultati presentati in questo capitolo è stata utilizzata una workstation dotata del seguente hardware:

CPU Intel® Core i7 950 @ 3.07 GHz, 12 GB di RAM

GPU NVIDIA® GeForce GTX 580, 1.5 GB di host memory

Tabella 5.1: Tempo di esecuzione medio del modulo RH_I per singola scansione TAC (dati ottenuti sul dataset LIDC).

	tempo medio (s)	guadagno (rispetto al precedente)	guadagno (assoluto)
CPU$_1$: autovalori iterativi	69.5	–	–
CPU$_2$: autovalori analitici	35.7	1.94	1.94
CPU$_3$: autovalori iterativi e criterio di L.C.	33.2	1.08	2.09
CPU$_4$: autovalori analitici e criterio di L.C.	27.8	1.19	2.5
GPU: implementazione CUDA	1.8	15.44	38.6

5.1 Tempi di esecuzione

In tabella 5.1 sono riportati i tempi medi per scansione TAC relativi agli algoritmi sviluppati su CPU e GPU. Un istogramma dei tempi di esecuzione è riportato in figura 5.1.

I valori riportati si riferiscono solo al tempo di esecuzione del modulo RH_I (algoritmo *dot-enhancer*) e non a tutta la catena del CAD. Inoltre non tengono conto del tempo necessario a caricare i dati da disco.

Nella prima riga della tabella 5.1 è riportato il tempo di esecuzione medio del modulo RH_I, implementato utilizzando l'algoritmo *dot-enhancer* non ottimizzato (CPU$_1$).

Utilizzando il calcolo analitico degli autovalori (seconda riga, CPU$_2$) il tempo di esecuzione viene dimezzato. Va ricordato che il calcolo degli autovalori, nell'implementazione CPU, non avviene per tutti i voxel della matrice, ma solo per quelli appartenenti al volume polmonare.

Nella terza (CPU$_3$) e quarta (CPU$_4$) riga della tabella 5.1 sono confrontati i tempi di esecuzione ottenuti utilizzando il criterio di Liénard-Chipart (teorema 1 a pagina 35) e i due metodi di calcolo degli autovalori. L'implementazione CPU$_3$ permette di ottenere un guadagno di un fattore 2.09, analoga al guadagno introdotto con l'utilizzo del calcolo analitico degli autovalori (CPU$_2$).

Utilizzando entrambe le ottimizzazioni (quarta riga, CPU$_4$) il guadagno ottenuto, rispetto all'implementazione originale, è di un fattore 2.5. Infatti, nonostante entrambe le ottimizzazioni apportino circa un fattore 2 rispetto

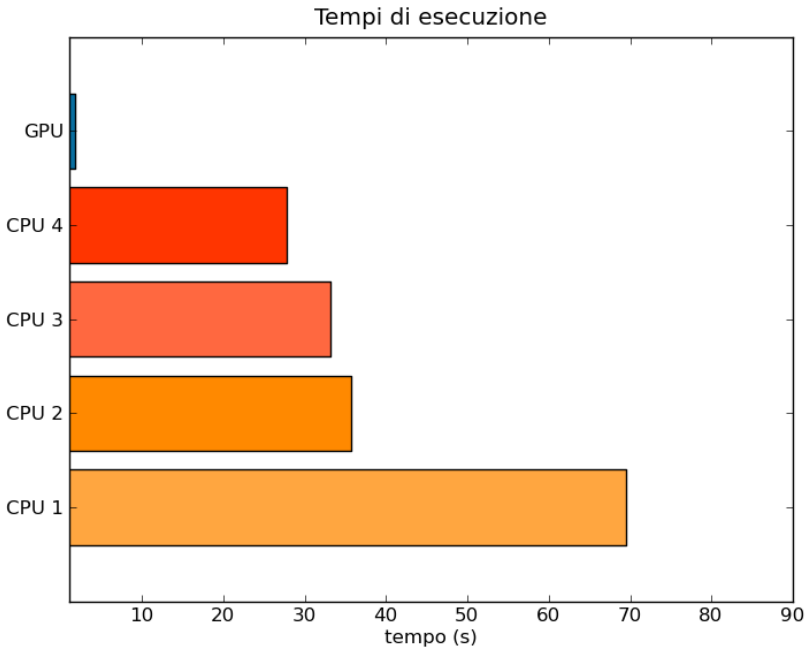


Figura 5.1: Confronto dei tempi di esecuzione del modulo RH_I al variare delle ottimizzazioni utilizzate.

all'originale, l'utilizzo del criterio di Liénard-Chipart porta a dover calcolare un numero molto minore di autovalori, da cui il minor guadagno globale, rispetto a quello prevedibile dal semplice prodotto dei guadagni ottenuti precedentemente.

Nell'ultima riga della tabella 5.1 è riportato il guadagno ottenuto con l'implementazione basata su GPU dell'algoritmo, descritta nel capitolo 4. La singola scansione viene analizzata in media in 1.8 secondi, ottenendo un guadagno rispetto alla versione originale dell'algoritmo di circa un fattore 40. Rispetto alla versione migliore basata su CPU (CPU_4), il guadagno è di un fattore 15.

Questo guadagno è tanto più notevole considerando che l'algoritmo basato su GPU deve calcolare (per motivi di efficienza) gli autovalori della matrice hessiana per ogni voxel dell'immagine: in questo modo esegue in media il calcolo di un fattore 100 di autovalori in più rispetto all'algoritmo basato su CPU.

Inoltre è da notare che il guadagno massimo su CPU è ottenibile solo utilizzando ottimizzazioni che rendono l'algoritmo non adatto ad utilizzi differenti da quelli dell'identificazione di strutture sferiche (come ad esempio l'iden-

tificazione di vasi). Con algoritmi basati su GPU è possibile implementare algoritmi generici mantenendo un elevato guadagno di prestazioni.

Un possibile svantaggio a cui si va incontro utilizzando le GPU, è dato dal fatto che i calcoli in virgola mobile sono eseguiti in singola precisione per ottenere il massimo della velocità. Questo, a seconda dell'applicazione, può essere un problema. È comunque un problema superabile poiché è comunque sempre possibile eseguire i calcoli in doppia precisione al prezzo di una perdita di prestazioni in termini di tempo di esecuzione, limitato a circa un fattore 2.

Nella sezione 5.3 sarà dimostrato come l'utilizzo della singola precisione, ai fini di questo sistema CAD, è sufficiente ad ottenere risultati analoghi a quelli ottenuti dalle implementazioni basate su CPU.

Per le successive valutazioni sono state utilizzate due differenti implementazioni: l'implementazione GPU e quella CPU completamente ottimizzata (CPU_4).

5.2 Curve di risposta

Per controllare la correttezza delle implementazioni dell'algoritmo *dot-enhancer* (su CPU e su GPU) è possibile confrontare la curva di risposta sperimentale con quelle teoriche ottenute nella sezione 2.4 (figure 2.11 e 2.12). Le curve di risposta sperimentali sono ottenute eseguendo l'algoritmo su immagini contenenti una sfera di profilo gaussiano (figura 2.10), valutando l'intensità della risposta del filtro al centro di questa sfera. Ogni punto della curva è ottenuto variando il parametro di scala della sfera campione, a parità dei parametri del filtro.

In figura (5.2a) e (5.2b) sono riportate le curve di risposta sperimentali per l'algoritmo basato su CPU e su GPU. Entrambe le curve riproducono fedelmente la curva di risposta teorica (eq. 2.15). L'algoritmo basato su GPU si discosta maggiormente dalla curva teorica per due motivi. Il primo motivo è dovuto al fatto che, mentre la CPU esegue i calcoli in doppia precisione, la GPU li esegue in singola precisione. A causa di ciò nell'intero calcolo si possono ottenere degli errori numerici di arrotondamento.

Il secondo motivo è dovuto all'utilizzo di differenti algoritmi di convoluzione nelle due implementazioni.

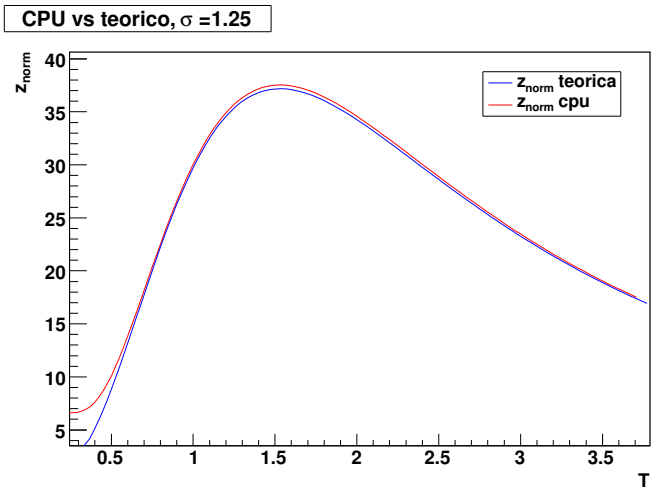
In figura 5.3a e 5.3b sono riportati gli andamenti sperimentali per la funzione z_{max} (eq. 2.16), per entrambi gli algoritmi.

Si può notare come i grafici sperimentali siamo in accordo con la curva teorica partendo da valori $\sigma \gtrsim 0.5$. Per valori di σ sotto questa soglia l'operazione di derivata non è ben definita. Infatti il campionamento della

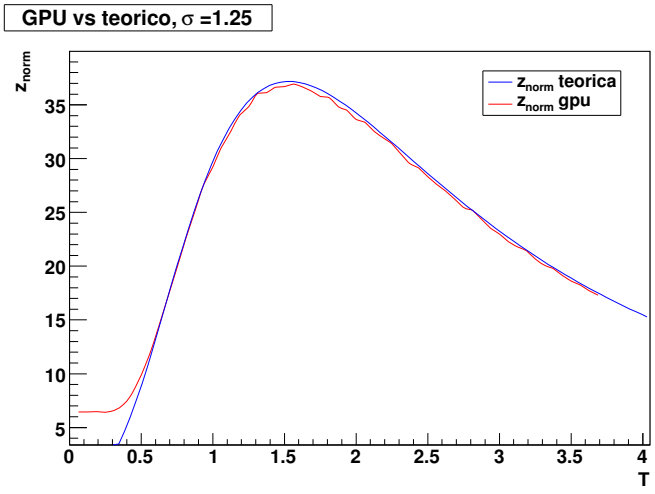
maschera gaussiana introduce artefatti numerici per cui la risposta del filtro non è più quella aspettata.

5.2.1 Test su immagini sintetiche

Per visualizzare il comportamento dell'algoritmo su immagini contenenti diverse forme geometriche è stata generata un'immagine campione con sfere e cilindri di profilo gaussiano, con vari parametri di scala. In figura 5.4a è riportato un *rendering* tridimensionale dell'immagine sintetica di input, e in figura 5.4b è riportato il *rendering* della matrice z_{max} (output del filtro). Si osserva come le strutture sferiche siano caratterizzate da valori elevati nella risposta del filtro, mentre gli oggetti cilindrici sono soppressi.

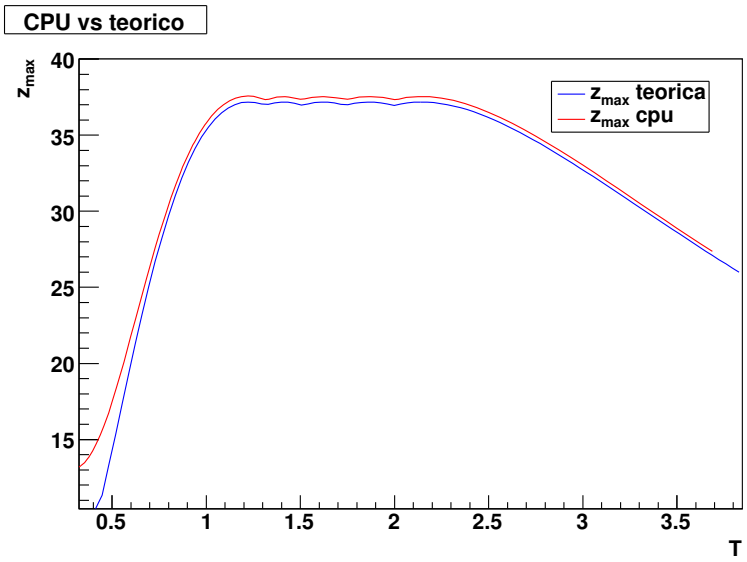


(a) CPU

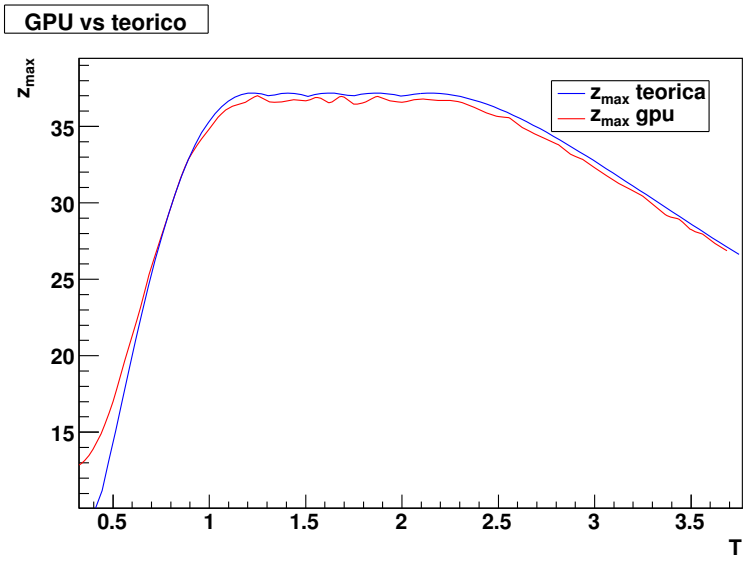


(b) GPU

Figura 5.2: Curva di risposta dell'algoritmo dot-enhancer per una singola scala (entrambe le implementazioni) a confronto con la curva teorica z_{norm} (eq. 2.15).

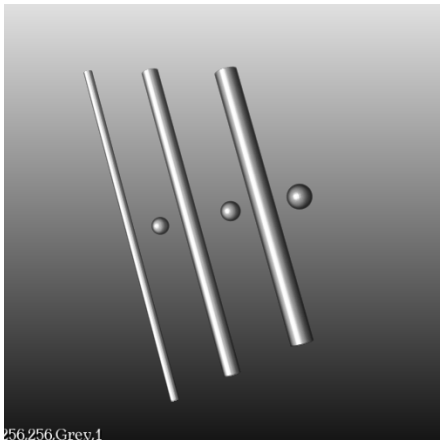


(a) CPU

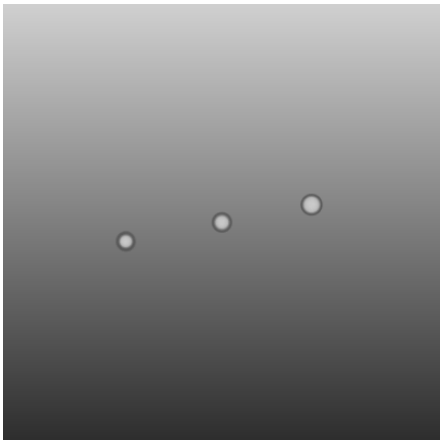


(b) CPU

Figura 5.3: Curva di risposta dell'algoritmo dot-enhancer multiscala (entrambe le implementazioni) a confronto con la curva teorica z_{max} (eq. 2.16).



(a) Immagine di input.



(b) Immagine di output dell'algoritmo *dot-enhancer*: valori della funzione z_{max}.

Figura 5.4: Risposta dell'algoritmo dot-enhancer a immagini sintetiche: a sinistra (a) è riportato un *volume-rendering* dell'immagine di input contenente sfere e cilindri di varie dimensioni (σ). A destra (b) è visibile il *volume-rendering* dell'output del filtro dot-enhancer, la matrice z_{max}.

5.3 Allenamento del sistema CAD

5.3.1 Descrizione del dataset

Le TAC utilizzate provengono dal database LIDC descritto al paragrafo 1.3.2 a pagina 5. Le 138 scansioni disponibili sono state suddivise in due insiemi, $\text{LIDC}_{\text{train}}$ e $\text{LIDC}_{\text{validation}}$, utilizzati rispettivamente per l'allenamento del CAD e per la validazione dell'allenamento. Le lesioni e i casi disponibili sono stati suddivisi in modo tale che i due insiemi contengano un ugual numero di casi e di noduli interni: ciascuno contiene 69 scansioni e 137 noduli interni.

Ogni annotazione è corredata dall'indicazione di quanti dei 4 radiologi hanno individuato la lesione. Selezionando quindi un valore di soglia $n \in [1, 4]$, è possibile definire come *rilevanti* gli oggetti identificati da almeno n radiologi, e con *irrilevanti* le restanti annotazioni. Il valore di soglia n seleziona quindi un Livello di Accordo (LA_n), identificando gli oggetti annotati da almeno n radiologi.

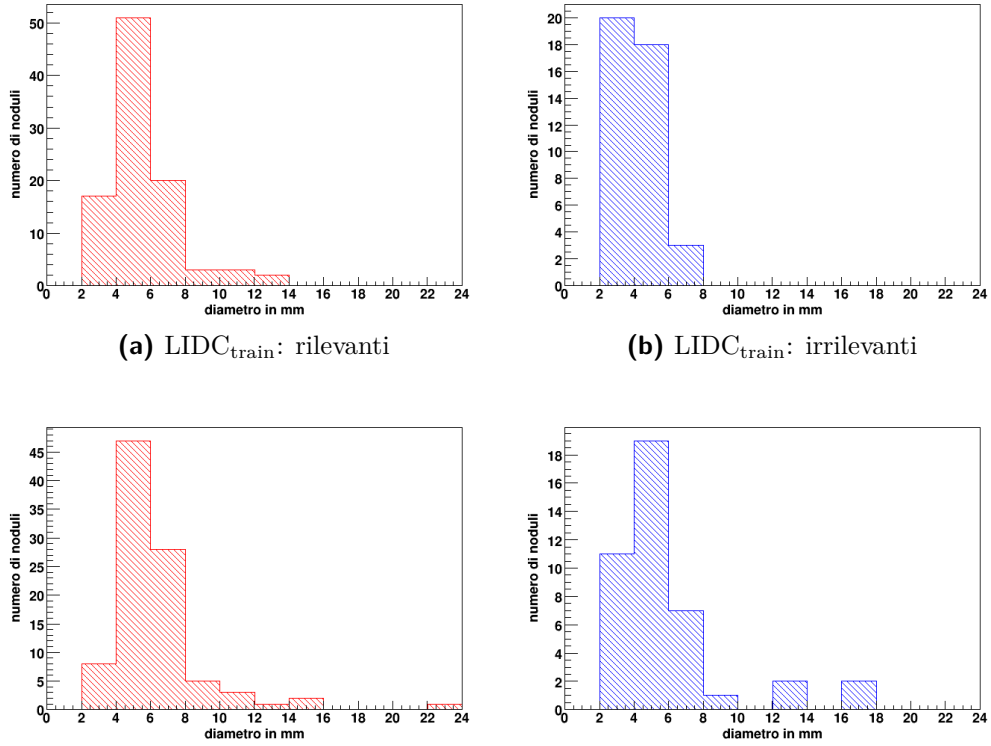
Nella fase di allenamento del CAD la soglia selezionata è $n = 2$. Il parametro è stato scelto empiricamente e permette di ottenere un buon numero di lesioni, 96 nell'insieme $\text{LIDC}_{\text{train}}$ e 95 nell'insieme $\text{LIDC}_{\text{validation}}$, mantenendo una buona soglia di confidenza sul rischio di introdurre, in fase di allenamento, esempi confermati da un solo radiologo. In figura 5.5 sono riportati i diametri dei noduli annotati nel dataset, suddivisi per tipo (rilevanti e irrilevanti) a questa soglia (LA_2). In tabella 5.2 è riportato, per entrambi i dataset, il numero di lesioni annotate al variare del LA.

5.3.2 Selezione dei parametri

Il modulo di identificazione dei candidati noduli interni si basa sull'algoritmo *dot-enhancer* (paragrafo 2.4 e schema 1 a pagina 34) e dipende da 3 parametri. I parametri di scala σ_{min} (minimo) e σ_{max} (massimo) con cui

Tabella 5.2: Numero di noduli interni presenti nelle annotazioni, al variare del livello di accordo.

LA	$\text{LIDC}_{\text{train}}$	$\text{LIDC}_{\text{validation}}$
1	137	137
2	96	95
3	66	71
4	44	34



(a) $\text{LIDC}_{\text{train}}$: rilevanti

(b) $\text{LIDC}_{\text{train}}$: irrilevanti

(c) $\text{LIDC}_{\text{validation}}$: rilevanti

(d) $\text{LIDC}_{\text{validation}}$: irrilevanti

Figura 5.5: Istogramma dei diametri dei referti del dataset LIDC con LA_2.

è effettuato lo *smoothing* gaussiano per il calcolo della matrice hessiana, e il numero n di passi intermedi utilizzato per effettuare l'analisi multi-scala. Sono stati effettuati vari test e i parametri che hanno dato migliori risultati sono stati $\sigma_{min} = 3.5$, $\sigma_{max} = 7$, con un numero di passi intermedi $n = 5$. In tabella 5.3 è riportato il numero delle lesioni non rilevate da questo modulo su entrambi i dataset disponibili con LA_2: entrambe le implementazioni hanno le medesime prestazioni.

Le liste di candidati noduli provenienti da questo modulo contengono migliaia di ROI per ciascuna scansione TAC. Per limitare il numero dei falsi positivi che dovranno essere successivamente classificati, queste liste sono tagliate in base al valore di output dell'algoritmo (z_{max}), che associa a ogni voxel un valore corrispondente alla probabilità di essere centro di una struttura sferica (sezione 2.2.5 a pagina 15).

In figura 5.6 sono riportati gli istogrammi dei valori di z_{max} per i candidati noduli identificati dal moduli RH_I^{GPU} corrispondenti ai noduli e ai falsi positivi. Il solo valore della funzione z_{max} non è sufficiente a distinguere le

Tabella 5.3: Numero di lesioni non rilevate per ciascun dataset, per entrambe le implementazioni dell'algoritmo *dot-enhancer*.

	lesioni non rilevate	
	$\text{LIDC}_{\text{train}}$	$\text{LIDC}_{\text{validation}}$
RH_I^{CPU}	5	2
RH_I^{GPU}	5	2

Tabella 5.4: Numero di lesioni identificate e numero di falsi positivi per scansione al variare del taglio sul valore di z_{max}, per entrambe le implementazioni RH_I e con LA_2.

z_{max}	RH_I^{GPU}		RH_I^{CPU}	
	sensibilità	FP/scansione	sensibilità	FP/scansione
0	1.00	2122.3	1.00	1650.6
5	1.00	1219.2	1.00	1178.8
10	0.99	402.2	0.99	410.1
15	0.98	154.9	0.99	161.7
20	0.95	71.1	0.95	74.5
25	0.90	36.9	0.90	38.6
30	0.87	19.7	0.88	20.9

due classi, ma un adeguato taglio permette di ridurre sensibilmente il numero dei falsi positivi da classificare. In tabella 5.4 sono riportati i risultati ottenuti a vari tagli del valore di z_{max} per entrambi i moduli.

Poiché è preferibile che il sistema CAD non presenti un elevato numero di falsi positivi per scansione, è stato scelto un taglio che permettesse di avere un basso numero di FP/scansione, anche a discapito della sensibilità massima. Il valore $z_{max} = 25$ è stato scelto come compromesso fra sensibilità e numero di FP/scansione.

5.3.3 Riduzione del numero dei falsi positivi

Attraverso il metodo VBNA (paragrafo 2.2.5), per ogni candidato nodulo rilevato dal modulo di ricerca dei noduli interni, vengono generati i vettori di features utilizzati per la classificazione e la riduzione del numero dei falsi positivi. Per la classificazione è stata utilizzata una Support Vector Machine (SVM) di tipo lineare. Un classificatore SVM è un classificatore di tipo

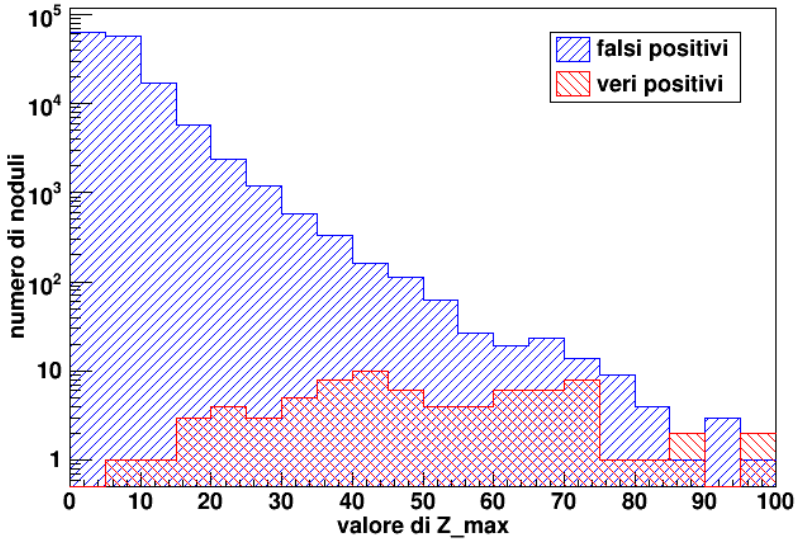


Figura 5.6: Istogrammi dei valori della funzione z_{max} per i candidati noduli identificati dal modulo RH_I^{GPU}, separati fra veri positivi e falsi positivi. Per una migliore visualizzazione sono stati riportati solo i candidati con valori di intensità $z_{max} \leq 100$.

supervisionato, ovvero necessita di un insieme di esempi su cui essere allenato. Idealmente un classificatore SVM produce un iperpiano che separa le classi dei vettori di features. Per permettere una certa flessibilità nella separazione in categorie, i modelli SVM lineari hanno un parametro di costo, C, che controlla il bilanciamento fra una separazione rigida e la possibilità di avere alcune classificazioni errate. All'aumentare del valore di C, aumenta il costo di una classificazione errata e viene forzata la creazione di un modello più accurato per i dati di allenamento.

Selezione del modello

Il requisito più critico in un sistema CAD è quello di riuscire a generalizzare, ovvero ad avere prestazioni analoghe anche su insiemi di dati non utilizzati per l'allenamento. È fondamentale quindi riuscire ad allenare il classificatore in modo che riesca a generalizzare correttamente dai dati disponibili. Poichè in linea di principio è possibile costruire un gran numero di classificatori dallo stesso insieme di dati di allenamento (ad esempio variando il parametro C), è necessaria una tecnica che permetta di valutare quale fra i vari classificatori sia ottimale.

Per fare ciò è stata utilizzata la tecnica della *cross-validation* [20], utilizzata comunemente per la selezione del modello migliore in termini di capacità di generalizzazione.

L'insieme dei dati di allenamento viene suddiviso in due sottoinsiemi: un sottoinsieme è utilizzato per l'allenamento del classificatore, l'altro è utilizzato per controllare le capacità di generalizzazione ottenute dal classificatore.

In particolare in questo caso è stata utilizzata la *Leave One Patient Out cross-validation* (LOPO) [39]. Dall'insieme di allenamento ($\text{LIDC}_{\text{train}}$) contenente 69 casi, 1 caso viene rimosso dall'insieme. Con i rimanenti 68 casi viene allenato il classificatore SVM lineare con un determinato parametro C, e le sue prestazioni valutate sul caso non utilizzato per l'allenamento. Il procedimento è ripetuto tante volte quanti sono i casi presenti nel dataset di allenamento.

Sono stati ottenuti quindi 69 classificatori che permettono una stima delle prestazioni del classificatore allenato sull'intero dataset e validato su un dataset di caratteristiche omogenee. Questo metodo è molto utile nel caso in cui non ci sia una grande disponibilità di dati su cui effettuare l'allenamento, dato che permette di utilizzare lo stesso dataset sia per l'allenamento che per la validazione.

La scelta del modello (C) migliore viene effettuata calcolando il valore dell'area sottesa dalla curva ROC (AUC) [40], ottenuta con il metodo LOPO. L'AUC possiede infatti un'importante proprietà statistica: è equivalente alla probabilità che un osservatore classifichi in modo più deciso un vettore di features, scelto a caso, corrispondente alla classe nodulo, rispetto a uno appartenente alla classe non nodulo (sempre scelto casualmente) [40].

Non essendoci indicazioni teoriche su quale possa essere il miglior valore per il parametro C sono stati utilizzati valori nell'intervallo $[2^{-20}, 2^{20}]$.

In figura 5.7 sono riportati gli andamenti del valore di AUC al variare del parametro C. Ad un maggior valore di AUC corrisponde una probabilità più elevata per il classificatore di avere una classificazione efficace [40]. Il massimo valore di AUC è stato ottenuto per valori di C prossimi a 1.

Per l'allenamento del modello è stato scelto quindi il valore $C = 1$ per entrambe le implementazioni del modulo di identificazione dei noduli interni.

5.3.4 Risultati LOPO sull'insieme $\text{LIDC}_{\text{train}}$

I vari classificatori ottenuti con il metodo LOPO a un C fissato possono essere utilizzati per avere una stima delle prestazioni che il sistema avrà su dati mai visti.

Ognuno dei 69 modelli infatti è stato allenato su un insieme complementare di dati e quindi possono essere utilizzati per ottenere una curva FROC del

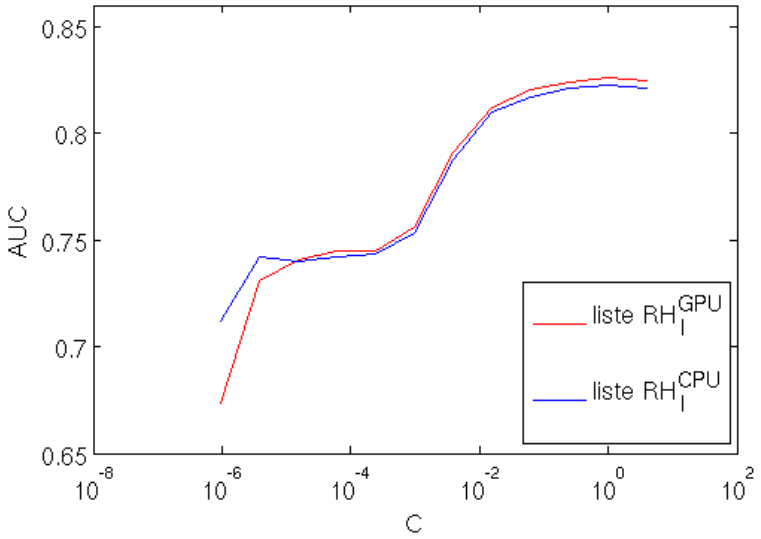


Figura 5.7: Valore dell'area sottesa dalla curva ROC (AUC), al variare del parametro C per entrambi i sistemi (CAD_I^{GPU} e CAD_I^{CPU}).

sistema sul dataset $\text{LIDC}_{\text{train}}$. Ovvero, selezionato il valore di C ottimale, si ottiene una famiglia di 69 classificatori, ciascuno dei quali è stato allenato su tutto il dataset tranne una scansione. Per ciascuna delle 69 scansioni, è disponibile quindi un classificatore che non è stato allenato su quei dati.

Con questo metodo è possibile dare una stima delle prestazioni che avrà il sistema CAD su un insieme di dati non utilizzato in allenamento, ovvero è possibile stimare le sue capacità di generalizzazione.

Il metodo VBNA classifica i singoli vettori di features come appatrenenti o no a una ROI patologica. Per passare dalla classificazione del singolo vettore di features alla classificazione di tutta la ROI, si utilizza il procedimento descritto nella sezione 2.2.5 a pagina 15: una ROI è classificata come lesione se la risposta media per i vettori di features appartenenti a quella regione è sopra una determinata soglia. Al variare del valore di soglia si ottiene la curva FROC.

Per confrontare quantitativamente le curve FROC è stato utilizzato il metodo proposto in [23]: viene effettuata la media dei valori di sensibilità ottenuta a 7 punti predefiniti di FP/scansione (1/8, 1/4, 1/2, 1, 2, 4, 8). Questo valore sarà riferito come *punteggio FROC*.

Inoltre al variare del livello di accordo dei radiologi (LA_i, con $i \in 1, \ldots 4$) si ottengono 4 diverse liste di lesioni e quindi 4 curve FROC. In figura 5.8 sono riportate le curve FROC ottenute con i modelli ottenuti tramite il metodo LOPO utilizzando le ROI ottenute dalle due implementazioni del modulo di

Tabella 5.5: Punteggio FROC ottenuto con il metodo LOPO e $C = 1$ per entrambe le implementazioni del modulo di ricerca dei noduli interni.

LA	punteggio FROC		Δ (%)
	CAD_I^{CPU}	CAD_I^{GPU}	
1	0.398	0.405	1.7
2	0.495	0.508	2.6
3	0.529	0.542	2.4
4	0.603	0.607	0.7

Tabella 5.6: Punteggio FROC ottenuto sul dataset $\text{LIDC}_{\text{validation}}$ con il modello SVM allenato sul dataset $\text{LIDC}_{\text{train}}$

LA	punteggio FROC		Δ (%)
	CAD_I^{CPU}	CAD_I^{GPU}	
1	0.445	0.445	0
2	0.497	0.495	0.4
3	0.541	0.538	0.5
4	0.517	0.505	2.3

ricerca dei noduli interni (RH_I^{CPU} e RH_i^{CPU}). In tabella 5.5 sono riportati i valori del punteggio FROC delle curve precedenti. Nella colonna Δ sono riportate le differenze percentuali fra i due sistemi a parità di LA.

5.3.5 Risultati sull'insieme $\text{LIDC}_{\text{validation}}$

Utilizzando tutti i dati provenienti dal dataset $\text{LIDC}_{\text{train}}$, sono stati allenati due classificatori SVM lineari, con parametro $C = 1$, per classificare i dati provenienti dalle due implementazioni del modulo RH_I.

Nonostante la cross-validation sia un buon metodo per la selezione del modello, non garantisce necessariamente prestazioni analoghe su dati mai visti [39]. L'unico test possibile per controllare la capacità di generalizzazione del sistema è il controllo diretto su un nuovo insieme di dati. I due sistemi CAD_I sono stati eseguiti su tutto il dataset $\text{LIDC}_{\text{validation}}$. In tabella 5.6 sono riportati i punteggi FROC ottenuti dai due sistemi, e in figura 5.9 sono riportate le rispettive curve FROC al variare del livello di accordo.

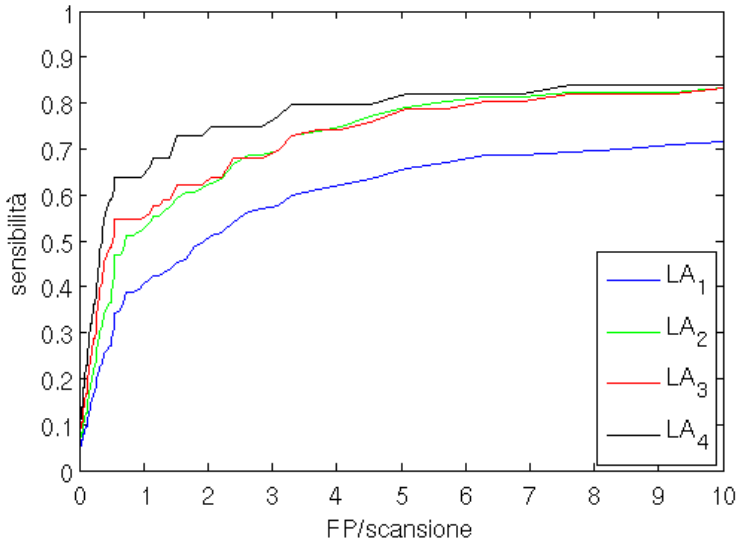
Tabella 5.7: Confronto dei punteggi FROC ottenuti in allenamento e in validazione per entrambi i sistemi (CAD_I^{CPU} e CAD_I^{GPU}) utilizzando le liste di annotazioni LA_2.

	LOPO $\text{LIDC}_{\text{test}}$	$\text{LIDC}_{\text{validation}}$	Δ (%)
CAD_I^{GPU}	0.508	0.495	2.6
CAD_I^{CPU}	0.495	0.497	0.04

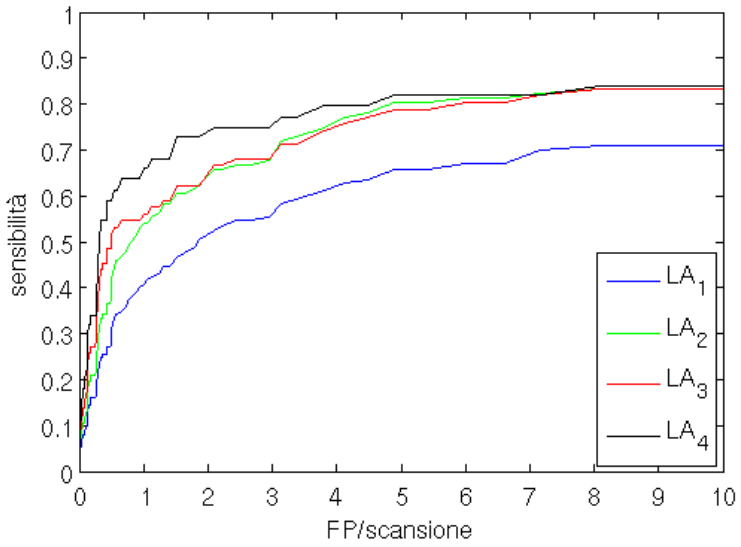
5.3.6 Osservazioni

Dalle tabelle 5.5 e 5.6 (quarta colonna) è possibile osservare come i due sistemi CAD ottengano, una volta allenati nelle medesime condizioni, le stesse prestazioni. Questo indica che le due differenti implementazioni non portano a risultati diversi sui dati reali. Questo dimostra come l'implementazione basata su GPU non diminuisca le prestazioni del sistema e ne aumenti le prestazioni in termini di tempo di esecuzione. In figura 5.10 sono confrontate graficamente le prestazioni dei due sistemi CAD, in modalità LOPO e sul dataset di validazione per LA_2. Anche dalla curva FROC è possibile notare come le due implementazioni non differiscano sensibilmente.

È possibile valutare le capacità di generalizzazione del sistema CAD sviluppato utilizzando i dati delle tabelle 5.5 e 5.6. Utilizzando i risultati ottenuti a LA_2 è stata ricavata la tabella 5.7. In questa tabella sono confrontati i punteggi FROC dei due sistemi cad sia in modalità allenamento (LOPO su $\text{LIDC}_{\text{train}}$) che validazione ($\text{LIDC}_{\text{validation}}$). Si nota come le differenze nei punteggi (colonna Δ) fra le due implementazioni siano minime. La procedura di selezione del modello è stata quindi capace di selezionare un set di parametri capace di una buona generalizzazione.

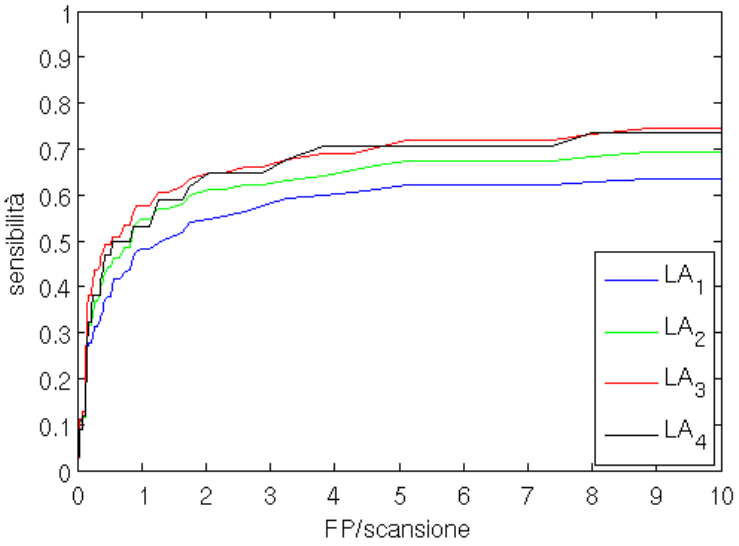


(a) CAD_I^{CPU}

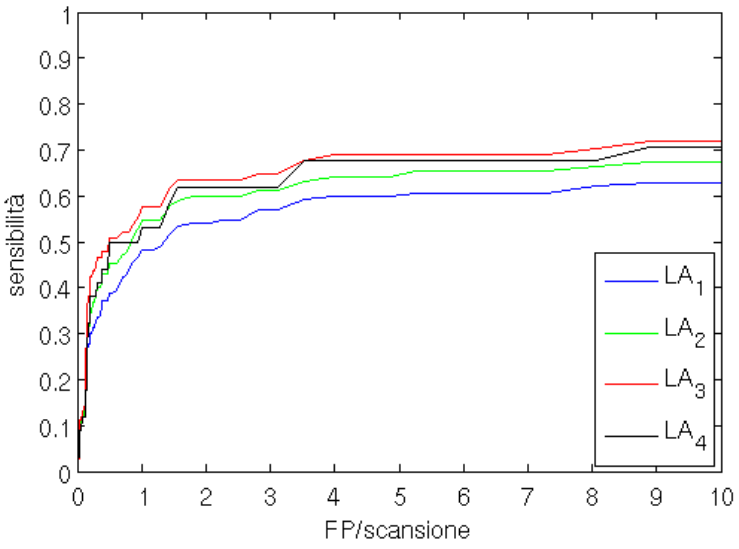


(b) CAD_I^{GPU}

Figura 5.8: Curve FROC corrispondenti al sistema CAD_I eseguito in modalità LOPO con parametro $C = 1$ relativo al dataset $\text{LIDC}_{\text{train}}$, al variare del livello di accordo (LA): implementazione CPU (a) e implementazione GPU (b).

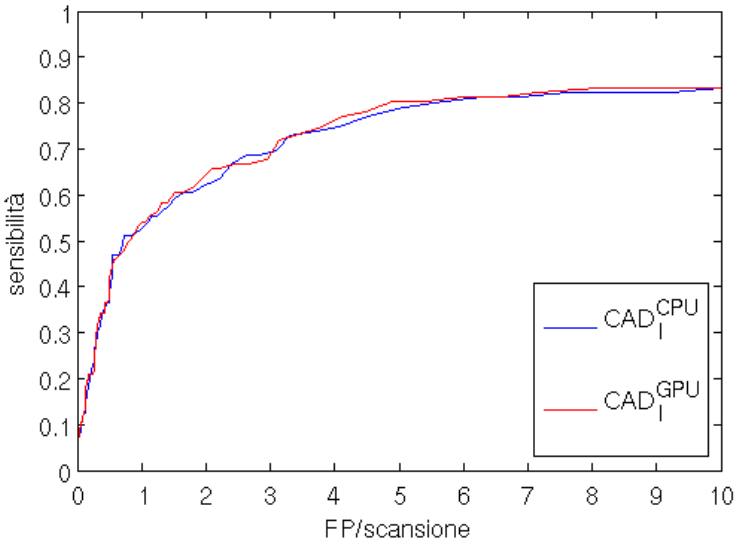


(a) CAD_I^{CPU}

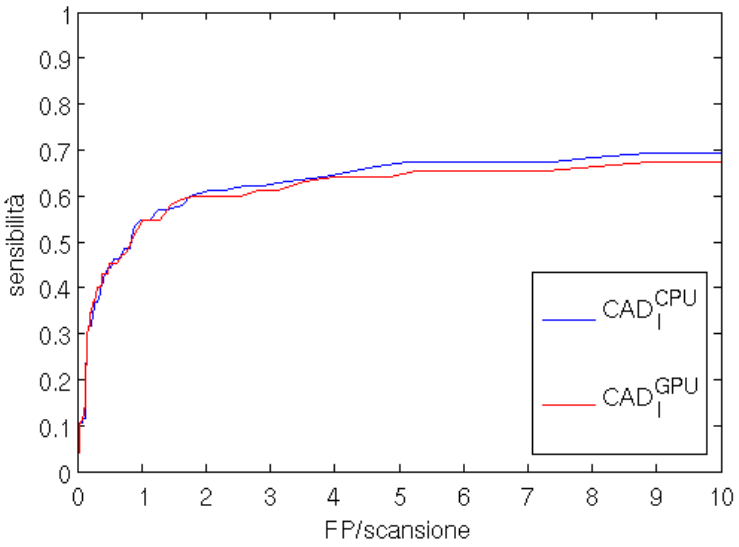


(b) CAD_I^{GPU}

Figura 5.9: Curve FROC corrispondenti al sistema CAD_I allenato sul dataset $\text{LIDC}_{\text{train}}$ ed eseguito sul dataset $\text{LIDC}_{\text{validation}}$, al variare del livello di accordo (LA): implementazione CPU (a) e implementazione GPU (b).



(a) curva FROC in modalità LOPO sull'insieme $\text{LIDC}_{\text{validation}}$



(b) curva FROC sull'insieme di validazione ($\text{LIDC}_{\text{validation}}$)

Figura 5.10: Confronto delle curve FROC relative ai due sistemi CAD_I in modalità LOPO (a) e in validazione (b) utilizzando le annotazioni con livello di accordo LA_2.

Conclusioni

In questo lavoro di tesi sono state valutate le possibilità di utilizzo di metodi di calcolo parallelo su *Graphical Processing Unit* (GPU), per la riduzione del tempo di esecuzione del sistema di *Computer Aided Detection* (CAD) sviluppato nell'ambito del gruppo di Pisa della collaborazione MAGIC-5 dell'INFN.

In particolare il lavoro si è concentrato sullo studio delle possibili ottimizzazioni, al fine di ridurre il tempo di esecuzione, dell'algoritmo deputato al riconoscimento automatico dei noduli interni in immagini da Tomografia Assiale Computerizzata (TAC) per la diagnosi precoce dei tumori polmonari.

In particolare sono state realizzate ottimizzazioni analitiche all'algoritmo implementato su CPU (RH_I^{CPU}) e successivamente è stata sviluppata una versione dell'algoritmo per GPU (RH_I^{GPU}).

Il guadagno ottenuto in termini di tempo di esecuzione è stato di un fattore 40 rispetto all'implementazione originale dell'algoritmo, e di un fattore 15 rispetto alla versione ottimizzata su CPU.

È stato inoltre dimostrato che il sistema CAD che utilizza il modulo RH_I^{GPU} (CAD_I^{GPU}) ha le stesse prestazioni in termini di capacità di identificare i candidati noduli rispetto al precedente sistema (CAD_I^{CPU}).

Il guadagno in termini di tempo di esecuzione (più di un ordine di grandezza) rende la GPU un ottimo candidato per l'implementazione di un sistema CAD più efficiente in termini di tempo di esecuzione. Questo potrà consentire un utilizzo più efficiente dei sistemi CAD sia nei programmi di *screening* della popolazione a rischio, sia durante la pratica clinica.

Bibliografia

[1] Ahmedin Jemal, Rebecca Siegel, Jiaquan Xu, and Elizabeth Ward. Cancer statistics, 2010. *CA: A Cancer Journal for Clinicians*, 60(5):277–300, September 2010.

[2] Cancer Facts & Figures Main. Website. `http://www.cancer.org/Research/CancerFactsFigures/index`.

[3] S Diederich, M G Lentschig, T R Overbeck, D Wormanns, and W Heindel. Detection of pulmonary nodules at spiral CT: comparison of maximum intensity projection sliding slabs and single-image reporting. *European Radiology*, 11(8):1345–1350, 2001. PMID: 11519541.

[4] Szabo E Doria-Rose VP. *Screening and prevention of lung cancer. In: Kernstine KH, Reckamp KL, eds. Lung cancer: a multidisciplinary approach to diagnosis and management.* New York: Demos Medical Publishing, 2010.

[5] National Lung Screening Trial Research Team. The national lung screening trial: Overview and study design. *Radiology*, 258(1):243 –253, January 2011.

[6] Reduced Lung-Cancer mortality with Low-Dose computed tomographic screening. *The New England Journal of Medicine*, June 2011. PMID: 21714641.

[7] Heidi C. Roberts, Demetris Patsios, Michael Kucharczyk, Narinder Paul, and Timothy P. Roberts. The utility of computer-aided detection (CAD) for lung cancer screening using low-dose CT. *International Congress Series*, 1281:1137–1142, May 2005.

[8] J. Anthony Seibert. X-Ray imaging physics for nuclear medicine technologists. part 1: Basic principles of X-Ray production. *Journal of Nuclear Medicine Technology*, 32(3):139 –147, 2004.

[9] Samuel G. Armato III, Geoffrey McLennan, Michael F. McNitt-Gray, Charles R. Meyer, David Yankelevitz, Denise R. Aberle, Claudia I. Henschke, Eric A. Hoffman, Ella A. Kazerooni, Heber MacMahon, Anthony P. Reeves, Barbara Y. Croft, Laurence P. Clarke, and For the Lung Image Database Consortium Research Group. Lung image database consortium: Developing a resource for the medical imaging research community1. *Radiology*, 232(3):739 –748, 2004.

[10] Michael F McNitt-Gray, 3rd Armato, Samuel G, Charles R Meyer, Anthony P Reeves, Geoffrey McLennan, Richie C Pais, John Freymann, Matthew S Brown, Roger M Engelmann, Peyton H Bland, Gary E Laderach, Chris Piker, Junfeng Guo, Zaid Towfic, David P-Y Qing, David F Yankelevitz, Denise R Aberle, Edwin J R van Beek, Heber MacMahon, Ella A Kazerooni, Barbara Y Croft, and Laurence P Clarke. The lung image database consortium (LIDC) data collection process for nodule detection and annotation. *Academic Radiology*, 14(12):1464–1474, December 2007. PMID: 18035276.

[11] 3rd Armato, Samuel G, Geoffrey McLennan, Drive Hawkins, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, Ella A Kazerooni, Heber MacMahon, Edwin J R Van Beeke, David Yankelevitz, Alberto M Biancardi, Peyton H Bland, Matthew S Brown, Roger M Engelmann, Gary E Laderach, Daniel Max, Richard C Pais, David P Y Qing, Rachael Y Roberts, Amanda R Smith, Adam Starkey, Poonam Batrah, Philip Caligiuri, Ali Farooqi, Gregory W Gladish, C Matilda Jude, Reginald F Munden, Iva Petkovska, Leslie E Quint, Lawrence H Schwartz, Baskaran Sundaram, Lori E Dodd, Charles Fenimore, David Gur, Nicholas Petrick, John Freymann, Justin Kirby, Brian Hughes, Alessi Vande Casteele, Sangeeta Gupte, Maha Sallamm, Michael D Heath, Michael H Kuhn, Ekta Dharaiya, Richard Burns, David S Fryd, Marcos Salganicoff, Vikram Anand, Uri Shreter, Stephen Vastagh, Barbara Y Croft, and Laurence P Clarke. The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans. *Medical Physics*, 38(2):915–931, February 2011. PMID: 21452728.

[12] Lung image database consortium. Accessed 16 december 2010. `https://wiki.nci.nih.gov/display/cip/lidc`.

[13] DICOM. informazioni disponibili su: `http://medical.nema.org/`.

[14] A. Geissb¨ uhler J. Heuberger and H. Müller. Lung CT segmentation for image retrieval using the Insight Toolkit (ITK). *Medical Imaging and Telemedicine (MIT)*, August 2005.

[15] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3 edition, August 2007.

[16] Qiang Li, Shusuke Sone, and Kunio Doi. Selective enhancement filters for nodules, vessels, and airway walls in two- and three-dimensional CT scans. *Medical Physics*, 30(8):2040–2051, August 2003. PMID: 12945970.

[17] David S Paik, Christopher F Beaulieu, Geoffrey D Rubin, Burak Acar, R Brooke Jeffrey, Judy Yee, Joyoni Dey, and Sandy Napel. Surface normal overlap: a computer-aided detection algorithm with application to colonic polyps and lung nodules in helical CT. *IEEE Transactions on Medical Imaging*, 23(6):661–675, June 2004. PMID: 15191141.

[18] Alessandra Retico, Francesco Bagagli, Niccolo Camarlinghi, Carmela Carpentieri, Maria Evelina Fantacci, and Ilaria Gori. A voxel-based neural approach (VBNA) to identify lung nodules in the ANODE09 study. In *Medical Imaging 2009: Computer-Aided Diagnosis*, volume 7260, pages 72601S–8, Lake Buena Vista, FL, USA, February 2009. SPIE.

[19] Niccolò Camarlinghi, Ilaria Gori, Alessandra Retico, Roberto Bellotti, Paolo Bosco, Piergiorgio Cerello, Gianfranco Gargano, Ernesto Lopez Torres, Rosario Megna, Marco Peccarisi, and Maria Evelina Fantacci. Combination of computer-aided detection algorithms for automatic lung nodule identification. *International Journal of Computer Assisted Radiology and Surgery*, July 2011.

[20] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition, July 1998.

[21] V Kecman. *Learning and Soft Computing, Support Vector Machines, Neural Networks and Fuzzy Logic Models*. MIT Press, 2001.

[22] I Gori, F Bagagli, M E Fantacci, A Preite Martinez, A Retico, I De Mitri, S Donadio, C Fulcheri, G Gargano, R Magro, M Santoro, and S Stumbo. Multi-scale analysis of lung computed tomography images. *Journal of Instrumentation*, 2(09):P09007, 2007.

[23] Bram van Ginneken, Samuel G. Armato III, Bartjan de Hoop, Saskia van Amelsvoort-van de Vorst, Thomas Duindam, Meindert Niemeijer, Keelin Murphy, Arnold Schilham, Alessandra Retico, Maria Evelina

Fantacci, Niccolo Camarlinghi, Francesco Bagagli, Ilaria Gori, Takeshi Hara, Hiroshi Fujita, Gianfranco Gargano, Roberto Bellotti, Sabina Tangaro, Lourdes Bolanos, Francesco De Carlo, Piergiorgio Cerello, Sorin Cristian Cheran, Ernesto Lopez Torres, and Mathias Prokop. Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: The anode09 study. *Medical Image Analysis*, 14(6):707 – 722, 2010.

[24] Temesguen Messay, Russell C. Hardie, and Steven K. Rogers. A new computationally efficient cad system for pulmonary nodule detection in ct imagery. *Medical Image Analysis*, 14(3):390 – 406, 2010.

[25] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Springer, 1 edition, December 1993.

[26] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, August 1984.

[27] Tony Lindeberg. On scale selection for differential operators. *8TH SCIA*, 1993.

[28] Wen-mei Hwu, K. Keutzer, and T. G Mattson. The concurrency challenge. *Design & Test of Computers, IEEE*, 25(4):312–320, August 2008.

[29] NVIDIA. *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide, version 4.0*. NVIDIA Corp, 2011.

[30] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 1 edition, February 2010.

[31] NVIDIA. *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*. NVIDIA Corp, 2007.

[32] NVIDIA. *Fermi Architecture White Paper*. NVIDIA Corp, 2009.

[33] R. Deriche. Fast algorithms for Low-Level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:78–87, January 1990. ACM ID: 81086.

[34] S. Barnett. A new formulation of the Liénard-Chipart stability criterion. *Proc. Cambridge Phil. Soc*, 70:269, 1971.

[35] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[36] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*. Dover Publications, June 1965.

[37] J. Kopp. Efficient Numerical Diagonalization of Hermitian 3×3 Matrices. *International Journal of Modern Physics C*, 19:523–548, 2008.

[38] NVIDIA. *CUDA C Best Practices Guide*. NVIDIA Corp, 2011.

[39] M. Dundar, G. Fung, L. Bogoni, M. Macari, A. Megibow, and B. Rao. A methodology for training and validating a cad system and potential pitfalls. *International Congress Series*, 1268:1010 – 1014, 2004. CARS 2004 - Computer Assisted Radiology and Surgery. Proceedings of the 18th International Congress and Exhibition.

[40] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.