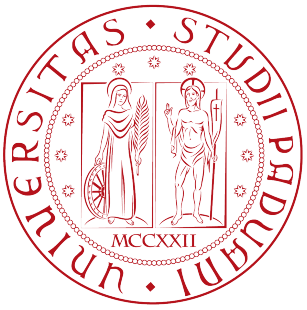


UNIVERSITÁ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

**Corso di Laurea Magistrale in**
**Ingegneria Informatica**

# *Ottimizzazione di un algoritmo per tomografia muonica*

*Relatore*
Prof. Giancarlo Calvagno

*Laureando*
Simone Ronchin

Padova, 18 Aprile 2011

# Abstract

Questa tesi nasce dallo sviluppo di un progetto del dipartimento di Fisica dell'Università di Padova che ha realizzato un prototipo funzionante [14] che realizza una tomografia muonica. Il software realizzato all'interno del progetto serviva per effettuare la ricostruzione dei contenuti analizzati con la tomografia in immagini facilmente consultabili. Il programma in questione però presenta tempi di esecuzione molto lunghi tempi e richiede una grande quantità di risorse hardware rendendo quindi l'effettiva applicazione della tomografia difficoltosa. Lo scopo di questo progetto è quindi quello di migliorare la qualità del software cercando di ridurre sensibilmente i tempi di ricostruzione senza aumentare la richiesta di risorse ne peggiorare il risultato al fine di rendere utilizzabile dal punto di vista pratico questa promettente tecnica di analisi.

Per ottenere i risultati proposti ci si è quindi concentrati sul codice già esistente al fine di individuarne tutte le inefficienze attraverso il supporto di tool di profiling per l'analisi precisa delle prestazioni e delle risorse utilizzate. Una volta individuate, grazie ai risultati precedenti, le situazioni più problematiche si sono proposte e implementate soluzioni che permettevano di ottenere un miglioramento delle prestazioni. I risultati ottenuti sono stati analizzati in dettaglio e confrontati con le prestazioni originali.

L'intero sviluppo ha portato ad un aumento delle prestazioni consistente fino quasi al dimezzamento dei tempi di esecuzione senza aumentare le risorse utilizzate nè peggiorare i risultati, inoltre si sono individuati altri punti di sviluppo per ottenere un possibile ulteriore miglioramento delle prestazioni.

# Indice


**1 Introduzione** **1**

**2 Tomografia muonica** **3**

2.1 Concetti di base . . . . . 3

2.2 Formulazione matematica . . . . . 5

2.2.1 Materiali omogenei . . . . . 5

2.2.2 Volumi non omogenei . . . . . 8

2.2.3 Gli errori di misurazione . . . . . 10

2.3 Ricostruzione ML/EM . . . . . 10

**3 Sviluppi presenti** **13**

3.1 Laboratorio dell'università di Padova . . . . . 13

3.2 Ricostruzione dei percorsi . . . . . 13

3.3 Ricostruzione tomografica . . . . . 14

3.4 Risultati . . . . . 15

3.4.1 Capacità di ricostruzione delle immagini . . . . . 16

3.4.2 Identificazione dei materiali . . . . . 16

3.5 Conclusioni . . . . . 17

**4 Obiettivi della tesi** **19**

**5 Pianificazione del lavoro** **21**

5.1 Studio del codice presente . . . . . 21

5.2 Creazione di una documentazione del codice . . . . . 21

5.3 Analisi del codice originario . . . . . 22
5.4 Scelta delle modifiche da effettuare . . . . . 22
5.5 Processo di modifica . . . . . 22
5.5.1 Modifica del codice . . . . . 23
5.5.2 Analisi prestazionale del codice modificato . . . . . 23
5.6 Commento al lavoro . . . . . 23

**6 Presentazione del codice attuale** **25**
6.1 Struttura del codice . . . . . 25
6.2 Funzionamento del software . . . . . 26
6.3 Tipologia di misure . . . . . 26
6.3.1 Primo tipo di campione . . . . . 27
6.3.2 Secondo tipo di campione . . . . . 29
6.4 Utilizzo della memoria RAM . . . . . 30
6.5 Utilizzo del processore . . . . . 31

**7 Tipologia di modifiche da effettuare** **35**
7.1 Primo livello di ottimizzazione . . . . . 36
7.2 Secondo livello di ottimizzazione . . . . . 36

**8 Risultati ottenuti** **39**
8.1 Primo livello di ottimizzazione . . . . . 39
8.1.1 Modifiche introdotte . . . . . 39
8.1.2 Risultati ottenuti . . . . . 40
8.1.3 Utilizzo della RAM . . . . . 45
8.1.4 Commento . . . . . 45
8.2 Secondo livello di ottimizzazione . . . . . 47
8.2.1 Modifiche introdotte . . . . . 47
8.2.2 Risultati ottenuti . . . . . 47
8.2.3 Utilizzo della RAM . . . . . 52
8.2.4 Commento . . . . . 52

**9 Sviluppi futuri** **55**

**10 Conclusioni** **57**

Capitolo 1

# Introduzione

La tomografia muonica nasce in seguito alla necessità di analizzare in modo non invasivo, in quanto il controllo manuale richiederebbe troppo tempo, insiemi di materiali etereogeni con lo scopo di individuare determinati tipi di materiali. In particolare nel nostro caso ci si pone l'obiettivo di realizzare una tomografia di container per la ricerca di materiale pericoloso (ad alto Z) come per esempio il piombo utilizzato per schermare materiale radioattivo per cercare di effettuarne uno smaltimento in modo illegale.

La tomografia muonica utilizza la particolarità dei raggi cosmici, infatti la maggior parte di essi che arrivano sulla Terra vengono fermati dall'atmosfera con interazioni che tipicamente producono una cascata di particelle secondarie, tra le quali sono presenti i muoni oggetto del nostro lavoro. A partire da una singola particella energetica, tali particelle possono arrivare fino alla superficie terrestre ed essere osservate con speciali apparecchiature. Quindi la loro alta penetrabilità all'interno dei materiali consente di poter analizzare ciò che le particelle attraversano nel loro percorso.

La prima applicazione che sfruttava i muoni come ispettori di materiali era basata sulla misura dell'assorbimento dei raggi cosmici per sondare l'interno di volumi altrimenti inaccessibili, il più famoso esempio di questa tecnica è senza dubbio quello effettuato nel 1971 dal premio Nobel Luis W. Alvarez [1], che esaminò la piramide di Chefren alla ricerca di sale nascoste. Un progetto basato sulla stessa tecnica è attivo ora all'Università di Napoli [2] con l'obiettivo di ottenere una radiografia del vulcano Vesuvio al fine di poter determinare lo sviluppo di un'eruzione qualora si

verificasse.

La tomografia muonica però si basa su un'altra particolarità di questa particella: la deviazione angolare che essi subiscono attraversando un materiale [3][4]. Quindi effettuando una misura della deviazione è possibile ottenere una ricostruzione della distribuzione del materiale in tre dimensioni, ottenendo un risultato che si può assimilare dal punto di vista abituale ad una TAC medica che sfrutta altri principi scientifici. Questo recente approccio richiede un numero inferiore di muoni e, di conseguenza, un tempo di attesa inferiore rispetto alla misura dell'assorbimento, ma richiede una strumentazione più complessa, dovendo misurare la direzione, e non solo la posizione, di ciascuna particella.

Il dipartimento di Fisica dell'Università di Padova ha realizzato un prototipo funzionante [14] che realizza una tomografia muonica e la tesi in questione va ad inserirsi in questo contesto. Il software realizzato per la ricostruzione ha ancora lunghi tempi di esecuzione e richiede una grande quantità di risorse hardware rendendo quindi l'analisi di grandi quantità di materiale estremamente lenta. Lo scopo è quindi quello di migliorare la qualità del software cercando di ridurre sensibilmente i tempi di ricorstruzione senza aumentare la richiesta di risorse al fine di rendere utiliazzabile dal punto di vista pratico questa promettente tecnica di analisi.

La tesi si svilupperà in questo modo:

- Presentazione dal punto di vista teorico della tomografia muonica.
- Esposizione di quanto già sviluppato da parte del dipartimento di fisica dell'Università di Padova.
- Analisi dettagliata dello scopo della tesi.
- Pianificazione del lavoro riguardante la tesi.
- Illustrazione del processo di ottimizzazione.
- Presentazione dei risultati ottenuti dal lavoro.
- Eventuali sviluppi successivi della tesi.

Capitolo 2

# Tomografia muonica

## 2.1 Concetti di base

La tomografia muonica è un nuovo tipo di tomografia basata sullo scattering dei raggi cosmici dei muoni. Questi provengono dalla spazio profondo, sono particelle stabili, per la maggior parte protoni, che bombardano continuamente la Terra. I muoni interagiscono con la materia per lo più attraverso la forza di Coulomb, non hanno interazioni nucleari e irradiano con molta meno facilità degli elettroni, inoltre perdono energia lentamente e solo attraverso interazioni elettromagnetiche. La deviazione dei muoni attraverso la materia dipende da diverse proprietà dei materiali ma il parametro dominante è il numero atomico del materiale attraversato. Ogni muone porta con se informazioni sugli oggetti che ha penetrato e misurando la deviazione di diversi muoni si possono individuare le proprietà degli oggetti attraversati. In particolare si possono individuare con facilità oggetti con numeri atomici elevati rispetto agli oggetti con numeri atomici bassi o medi[5][6][7].

Il concetto della tomografia è illustrato nella figura 2.1.

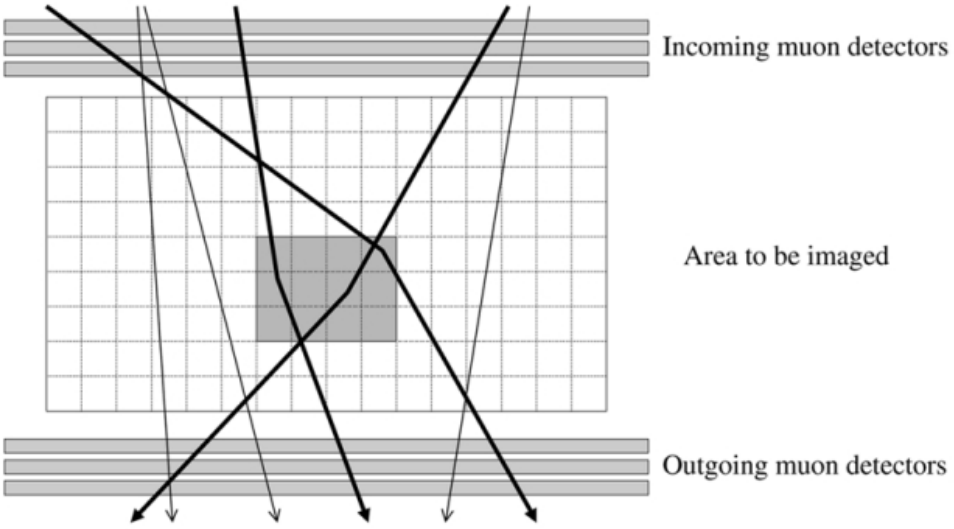


Figura 2.1: Concetto di tomografia muonica

Due o più piani sensibili ai muoni sono posizionati attorno al volume da analizzare, questi rilevatori misurano la posizione dei muoni in due coordinate ortogonali. I muoni che attraversano il volume da analizzare sono deviati a seconda del materiale di cui è composto il volume.

La ricostruzione discreta del volume è effettuata a partire dai dati che vengono forniti da diversi muoni. Viene usato l'algoritmo iterativo di massimizzazione dell'aspettazione (EM) per trovare la verosimiglianza massima per la densità degli oggetti in analisi.

La misura dell'angolo di deviazione dei muoni è stocastica, con media uguale a zero e deviazione standard che è definita dalle proprietà del materiale attraversato. I muoni non provengono da un definito numero discreto di direzioni ma la loro direzione è variamente distribuita attorno allo zenith. Infine le traiettorie dei muoni non sono rettilinee qualora incontino materiali diversi nel loro percorso.

## 2.2 Formulazione matematica

### 2.2.1 Materiali omogenei

Un raggio cosmico di muoni che attraversa un materiale effettua diverse deviazioni di traiettorie dovute alla forza di Coulomb. La traiettoria di uscita del muone è caratterizzata dalla variazione dell'angolo ($\Delta\theta$) e di posizione ($\Delta x$). La deviazione dell'angolo generalmente è di poche decine di mrads (1 mrad≈0.06°).

La distribuzione del 98% degli angoli può essere approssimata con una Gaussiana a media zero[8]:

$$f_{\Delta\theta}(\Delta\theta) \cong \frac{1}{\sqrt{2\pi}\sigma_\theta} \exp\left(-\frac{\Delta\theta^2}{2\sigma_\theta^2}\right). \tag{2.1}$$

La deviazione standard della distribuzione può essere espressa approssimativamente in termini della proprietà dei materiali, la più semplice è stata proposta da Rossi [12]

$$\sigma_\theta \cong \frac{15MeV}{\beta cp}\sqrt{\frac{H}{L_{rad}}} \tag{2.2}$$

dove $p$ è il momento di particella misurato in MeV/c, $H$ è la profondità del materiale e $L_{rad}$ è la lunghezza di radiazione del materiale. $\beta c$ è la velocità e assumiamo che $\beta$ sia uguale ad 1. La lunghezza di radiazione decresce con il numero atomico e la densità del materiale.

Stabiliamo un momento di muone nominale e definiamo la *scattering density* del materiale che ha una lungezza di radiazione, $L_{rad}$, come:

$$\lambda\left(L_{rad}\right) \equiv \left(\frac{15}{p_0}\right)^2 \frac{1}{L_{rad}}. \tag{2.3}$$

La scattering density $\lambda$ di un materiale rappresenta lo scarto quadratico medio dell'angolo dei muoni con momento nominale che attraversano un'unità di tale materiale.

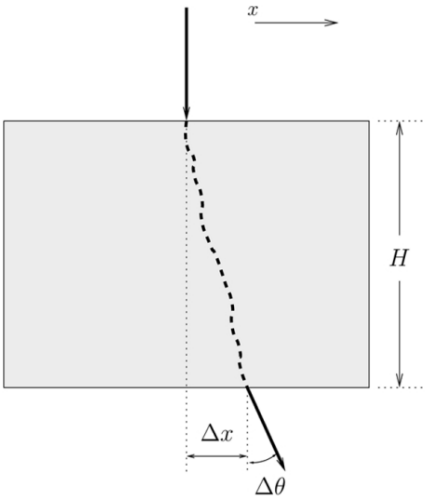


Figura 2.2: Proiezione in due dimensioni che mostra la variazione del percorso di un muone.

La varianza dello scattering di un muone con scattering density $\lambda$ e profondità H è:

$$\sigma_\theta^2 = \lambda H \left(\frac{p_0}{p}\right)^2 \tag{2.4}$$

ed essendo

$$p_r^2 = \left(\frac{p_0}{p}\right)^2 \tag{2.5}$$

abbiamo che:

$$\sigma_\theta^2 = \lambda H p_r^2. \tag{2.6}$$

Lo spostamento $\Delta x$ è correlato con l'angolo di scattering $\Delta\theta$, la distribizione dell'angolo di scatterting e della distanza è caratterizzata da una Gaussiana congiunta con media zero[8] e parametri:

$$\sigma_{\Delta x} = \frac{H}{\sqrt{3}}\sigma_{\Delta\theta} \tag{2.7}$$

$$\rho_{\Delta\theta\Delta x} = \frac{\sqrt{3}}{2}. \tag{2.8}$$

Esprimiamo la matrice di covarianza come:

$$\Sigma \equiv \begin{bmatrix} \sigma_{\Delta\theta}^2 & \sigma_{\Delta\theta\Delta x} \\ \sigma_{\Delta\theta\Delta x} & \sigma_{\Delta x}^2 \end{bmatrix} = \lambda \begin{bmatrix} H & \frac{H^2}{2} \\ \frac{H^2}{2} & \frac{H^3}{3} \end{bmatrix} p_r^2. \tag{2.9}$$

Quindi definendo:

$$A \equiv \begin{bmatrix} H & \frac{H^2}{2} \\ \frac{H^2}{2} & \frac{H^3}{3} \end{bmatrix} \tag{2.10}$$

abbiamo:

$$\Sigma = \lambda A p_r^2. \tag{2.11}$$

In tre dimensioni dobbiamo considerare lo scattering anche della coordinata $y$ ortogonale a $x$ e riferirci agli angoli di scattering $\Delta\sigma_x$ e $\Delta\sigma_y$ e spostamenti $\Delta x$ e $\Delta y$. Inoltre dobbiamo considerare la lunghezza del percorso tridimensionale e disporre di misure in un piano ortogonale al percorso del muone, come si può vedere in figura 2.3.

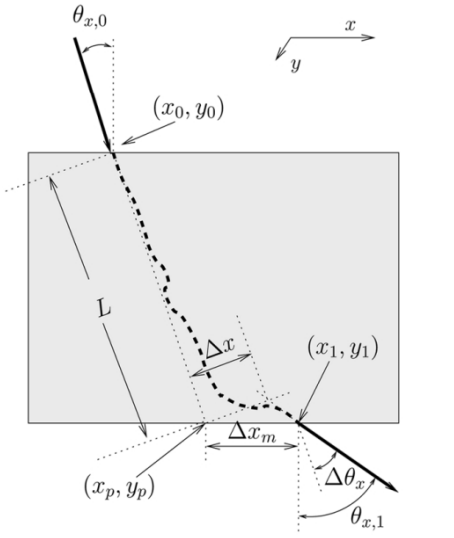


Figura 2.3: Parametri usati per la correzione al modello tridimensionale.

L'estensione della retta della traiettoria non deviata attraverso il primo piano e proiettata nel secondo definisce il punto $xp$ e $yp$ e la lunghezza di questo segmento $L$ è:

$$L = H\sqrt{1 + \tan^2\theta_{x,0} + \tan^2\theta_{y,0}} \equiv HL_{x,y}. \tag{2.12}$$

Definiamo la posizione $x$ di uscita del muone e l'angolo come $(x_1, \theta_{x,1})$ e quindi:

$$\Delta\theta_x = \theta_{x,1} - \theta_{x,0}. \tag{2.13}$$

La misura dello spostamento può essere calcolata come $x_m = x_1 - x_p$ ma la misura va ruotata nel piano ortogonale alla traiettoria iniziale del muone e aggiustata

per la lunghezza del percorso nelle tre dimensioni, quindi:

$$\Delta_x = (x_1 - x_p)\cos(\theta_{x,0})\, L_{xy} \frac{\cos(\theta_x - \theta_{x,0})}{\cos(\theta_x)}. \tag{2.14}$$

Ridefiniamo la matrice $A$:

$$A \equiv \begin{bmatrix} L & \frac{L^2}{2} \\ \frac{L^2}{2} & \frac{L^3}{3} \end{bmatrix}. \tag{2.15}$$

Il procedimento di lavoro sulla coordinata $y$ è uguale a quello della coordinata $x$ e le misure sono fatte in modo indipendente nelle due coordinate.

### 2.2.2 Volumi non omogenei

Se il volume non è omogeneo la situazione si modifica e rappresentiamo il profilo di densità in termini di una combinazione lineare di funzioni base tridimensionali $\{\phi_1, \ldots, \phi_j, \phi_N\}$ con coefficienti $\{v_1, \ldots, v_j, v_N\}$

$$\lambda(x, y, z) = \sum_j v_j \phi_j(x, y, z). \tag{2.16}$$

Useremo $\lambda_j$ per denotare il coefficiente della j-esima funzione.

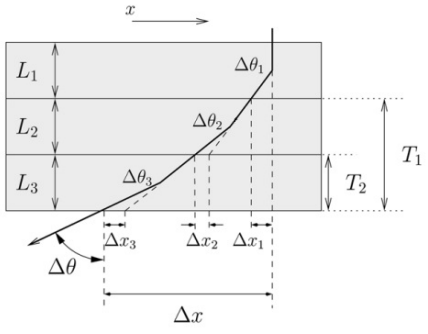


Figura 2.4: Lo scattering attraverso diversi tipi di materiale

Dividiamo l'area di lavoro in voxel, come per esempio in figura 2.4, ognuno con la propria informazione di $\Delta\theta$ e $\Delta x$. Gli scattering nascosti nel $j$-esimo voxel sono denotati come $\Delta\theta_j$ e come $\Delta x_j$.

Quindi possiamo andare a definire il $\Delta\theta$ totale su $\mathcal{N}$ voxel come:

$$\Delta\theta = \sum_{j\in\mathcal{N}} \Delta\theta_j \tag{2.17}$$

mentre il $\Delta x$ come:

$$\Delta x = \sum_{j\in N} (\Delta x_j + T_j \Delta\theta_j) \tag{2.18}$$

dove $T_j$ è definito come il percorso tridimensionale dal punto di uscita dal $j$-esimo voxel al punto di uscita del volume totale di ricostruzione.

Ora possiamo esprimere la covarianza aggregata dell'angolo di scattering e distanza per l'$i$-esimo muone e il $j$-esimo voxel:

$$\Sigma_{ij} = \lambda_j A_{ij} p_{r,i}^2 \tag{2.19}$$

dove:

$$A_{ij} \equiv \begin{bmatrix} L_{ij} & \frac{L_{ij}^2}{2} \\ \frac{L_{ij}^2}{2} & \frac{L_{ij}^3}{3} \end{bmatrix} \tag{2.20}$$

e $L_{ij}$ è la lunghezza del percorso dell'$i$-esimo muone attraverso il $j$-esimo voxel, ed è definito pari a zero per i voxel non attraversati dal raggio.

Combinando le equazioni (2.17)-(2.20) si ottiene

$$\Sigma_i = p_{r,i}^2 \sum_{j\leqslant N} \lambda_i W_{ij}, \tag{2.21}$$

dove $N$ è il numero dei voxel e la matrice del peso è definita da:

$$W_{ij} \equiv \begin{bmatrix} L_{ij} & \frac{L_{ij}^2}{2} + L_{ij}T_{ij} \\ \frac{L_{ij}^2}{2} + L_{ij}T_{ij} & \frac{L_{ij}^3}{3} + \frac{L_{ij}^2}{2}T_{ij} + L_{ij}T_{ij}^2 \end{bmatrix}. \tag{2.22}$$

Definiamo il vettore dei dati:

$$D_i \equiv \begin{bmatrix} \Delta\theta_i \\ \Delta x_j \end{bmatrix} \tag{2.23}$$

e denoteremo con $D$ tutte le misure da $M$ muoni.

Scriviamo il likelihood della densità $\delta$ come:

$$P(D|\lambda) = \prod_{i\leqslant M} P(D_i|\lambda) \tag{2.24}$$

con il fattore:

$$P\left(D_i|\lambda\right) = \frac{1}{2\pi|\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} D_i^T \Sigma_i^{-1} D_i\right). \tag{2.25}$$

### 2.2.3 Gli errori di misurazione

La posizione e l'angolo di ingresso di ingresso, così come quelli di uscita, misurati possono presentare degli errori di misurazione dovuti alle apparecchiature di rilevazione. Quindi questi si propagano lungo tutti gli angoli e distanze di scattering calcolate successivamente. Per evitare di influenzare i dati da questi errori di misurazione siamo costretti ad inserire una matrice di errore nei nostri calcoli al fine di tenere conto di questa possibilità:

$$E \equiv \begin{bmatrix} e^2_{\Delta\theta} & e_{\Delta\theta\Delta x} \\ e_{\Delta\theta\Delta x} & e^2_{\Delta x} \end{bmatrix}. \tag{2.26}$$

Nel nostro caso questa matrice si somma direttamente alla matrice di covarianza:

$$\Sigma_i = E + p^2_{r,i} \sum_{j \leqslant N} \lambda_i W_{ij}. \tag{2.27}$$

## 2.3 Ricostruzione ML/EM

Nell'applicazione in esame i dati osservati $D = \{D_i : 1 \leq i \leq M\}$ sono lo scattering misurato.

I dati nascosti $H = \{H_{ij} : 1 \leq i \leq M, 1 \leq j \leq N\}$ sono lo scattering, sia di angolo che di posizione, dell'$i$-esimo muone nel $j$-esimo voxel:

$$Q_{DLR} = E_{H|D,\lambda^{(n)}}\left[\log\left(P\left(D, H|\lambda\right)\right)\right]. \tag{2.28}$$

Questa funzione è il valore atteso del logaritmo del likelihood sia dei dati osservati che di quelli nascosti, dati il vettore dei parametri $\lambda$ che rispetta le distribuzioni condizionali di $H$ dato $D$ e il vettore di parametri $\lambda^{(n)}$.
Ogni iterazione dell'algoritmo consiste dei seguenti due passi:

- Primo passo: stima della distribuzione condizionale dei dati nascosti.
- Secondo passo: massimizzazione della funzione ausiliaria $Q$ che è il valore atteso della distribuzione caratterizzata al passo precedente.

In questo caso visto che i dati nascosti determinano i dati osservati la funzione ausiliaria diventa:

$$Q\left(\lambda;\lambda^{(n)}\right)=E_{H|D,\lambda^{(n)}}\left[\log\left(P\left(D,H|\lambda\right)\right)\right]. \tag{2.29}$$

A partire dal parametro stimato $\lambda^{(n)}$ un'iterazione dell'algoritmo produce il nuovo valore stimato $\lambda^{(n+1)}$ da:

$$\lambda^{(n+1)}=\arg\max_{\lambda}Q\left(\lambda;\lambda^{(n)}\right). \tag{2.30}$$

Formuliamo la distribuzione di probabilità dello scattering di un singolo muone attraverso un singolo Voxel seguendo semplicemente il modello statistico di un singolo piano:

$$P\left(H_{ij}|\lambda\right)=\frac{1}{2\pi|\Sigma_i|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}H_{ij}^{T}\Sigma_{ij}^{-1}H_{ij}\right). \tag{2.31}$$

La probabilità di un set di dati nascosti è il prodotto della probabilità di ogni elemento, quindi il logaritmo del likelihood può essere scritto come:

$$\log\left(P\left(D,H|\lambda\right)\right)=\sum_{j\leq N}\sum_{i:L_{ij}\neq0}\left(-\log\lambda_j-\frac{H_{ij}^{T}A_{ij}^{-1}H_{ij}}{2\lambda_j p_{r,i}^{2}}\right)+C \tag{2.32}$$

dove C rappresenta i termini che non contengono $\lambda$. Possiamo scrivere la funzione Q in questo modo quindi:

$$Q\left(\lambda;\lambda^{(n)}\right)=C+\sum_{j\leq N}Q_j\left(\lambda_j;\lambda_j^{(n)}\right) \tag{2.33}$$

dalla quale deriviamo:

$$Q\left(\lambda;\lambda^{(n)}\right)=-M_j\log\lambda_j-\frac{1}{2\lambda_j}\sum_{i:L_{ij}\neq0}S_{ij}^{(n)}. \tag{2.34}$$

In questa relazione $M_j$ sono il numero di raggi per i quali $L_{ij}\neq0$, ossia il numero di raggi che colpiscono il $j$-esimo voxel e $S_{ij}^{(n)}$ è definito come:

$$S_{ij}^{(n)}\equiv E_{H|D,\lambda^{(n)}}\left[p_{r,i}^{-2}H_{ij}^{T}A_{ij}^{-1}H_{ij}\right]. \tag{2.35}$$

Posta a zero la derivata rispetto a $\lambda_j$ dell'equazione (2.34), individuiamo la funzione iterativa per massimizzare la funzione ausiliaria:

$$\lambda_j^{(n+1)}=\frac{1}{2M_j}\sum_{i:L_{ij}\neq0}S_{ij}^{(n)}. \tag{2.36}$$

La forma quadratica di $S_{ij}$ garantisce la positività di $\lambda^{(n+1)}$.

Rimane da calcolare il valore atteso condizionato di $S_{ij}$. Sia $X$ la variabile aleatoria $H_{ij}|D_i$. Il valore atteso della forma quadratica $X^T A^{-1} X$ è:

$$E_{X^T A^{-1} X} = Tr\left(A^{-1}\Sigma_X\right) + \mu_X^T A^{-1} \mu_X \tag{2.37}$$

dove $\mu_X$ è la media e $\Sigma_X$ è la varianza di $X$.

Considerando che $D_i$ dipende linearmente da $H_{ij}$, esse risultano congiuntamente Gaussiane. La distribuzione condizionata di $H_{ij}$ dato $D_i$ è anch'essa Gaussiana, (applicando la teoria della distribuzioni multivariate) e considerando che $H_{ij}$ e $D_i$ hanno media zero, troviamo che:

$$\mu_X = \Sigma^T_{D_i H_{ij}} \Sigma^{-1}_{D_i} D_i \tag{2.38}$$

$$\Sigma_X = \Sigma_{H_{ij}} - \Sigma^T_{D_i H_{ij}} \Sigma^{-1}_{D_i} \Sigma_{D_i H_{ij}}. \tag{2.39}$$

Ricavando $\Sigma_{D_i}$ dall'equazione (2.27) e $\Sigma_{H_{ij}}$ dalla (2.19), possiamo scrivere la covarianza congiunta dei dati osservati $\Sigma_{D_i}$ e di quelli nascosti $\Sigma_{H_{ij}}$ attraverso una calcolo matriciale e mostrare che:

$$\Sigma_{D_i H_{ij}} A^{-1}_{ij} \Sigma^T_{D_i H_{ij}} = W_{ij} \left(p^2_{r,i} \lambda_j\right)^2. \tag{2.40}$$

Sostituendo i risultati da (2.37) e (2.40) nella (2.35) troviamo che:

$$\begin{aligned} S^{(n)}_{ij} &= p^{-2}_{r,i} Tr\left(A^{-1}_{ij}\Sigma_{H_{ij}} - A^{-1}_{ij}\Sigma^T_{D_i H_{ij}}\Sigma^{-1}_{D_i}\Sigma_{D_i H_{ij}}\right) \\ &\quad + p^{-2}_{r,i} D_i^T \Sigma^{-1}_{D_i} W_{ij} \Sigma^{-1}_{D_i} D_i \left(p^2_{r,i}\lambda^{(n)}_j\right)^2 = \\ &= 2\lambda^{(n)}_j + \left(D_i^T \Sigma^{-1}_{D_i} W_{ij} \Sigma^{-1}_{D_i} - Tr\left(\Sigma^{-1}_{D_i} W_{ij}\right)\right) \times p^2_{r,i} \left(\lambda^{(n)}_j\right)^2 \end{aligned} \tag{2.41}$$

Infine per considerare entrambe le coordinate usiamo semplicemente la media:

$$S^{(n)}_{ij} = \frac{S^{(n)}_{ij,x} + S^{(n)}_{ij,y}}{2}. \tag{2.42}$$

Capitolo 3

# Sviluppi presenti

Degli studi effettuati precedentemente si è occupato anche il dipartimento di fisica dell'università di Padova che ha deciso di rendere lo studio operativo, con un sistema di tomografia effettivamente funzionante.

## 3.1 Laboratorio dell'università di Padova

Il prototipo è stato messo in opera nel laboratorio di Legnaro (Padova, Italia) [14], questo sistema permette di analizzare un volume di circa 11 $m^3$, tale scelta è stata fatta per poter confrontarsi con una taglia di tomografia che possa avere anche una qualche applicazione reale.
Il sistema è costituito da due piani che misurano la presenza dei muoni, uno posto sopra ed uno sotto il volume da analizzare. I due misuratori hanno dimensioni di 300x200 $cm$ e un'altezza di 29 $cm$.

## 3.2 Ricostruzione dei percorsi

Il primo passo della ricostruzione dei percorsi è l'analisi dei pattern che identificano i luoghi che lo stesso muone colpisce lungo la propria traiettoria.
Data la scarsa quantità di muoni che giungono al rilevatore, per la maggior parte dei casi si ha un solo raggio per volta all'interno del volume, quindi la ricostruzione è facilitata e meno critica che in altri tipi di tomografie. Il secondo passo è la ricostruzione della traiettoria.

## 3.3 Ricostruzione tomografica

La ricostruzione del materiale che si trova nel volume tra i due rilevatori è basato sulla misurazione dello scattering dell'angolo che viene influenzato dalle proprietà del materiale attraversato, come spiegato nel capitolo precedente.

Nel processo di ricostruzione il volume contenuto tra i due rilevatori è suddiviso in $N$ voxel cubici, l'obiettivo è stimare la media dello scattering di ogni voxel. Questi voxel sono numerati con un singolo indice $j$, quindi il set di ricostruzione restituirà un set di $\lambda$ che inizialmente sono sconosciuti $\{\lambda_j; j = 1, \ldots, N\}$.

La procedura di ricostruzione utilizzata consiste nell'algoritmo iterativo Maximum Log-Likelihood ottimizzato. L'intera procedura di ricostruzione usata nello studio è basata sulla strategia List-Mode che consiste essenzialmente nell'elaborare le hits una alla volta anziché sistemarle in gruppi di eventi simili in un istogramma e analizzare la distribuzione dell'istogramma. La casualità dei raggi cosmici che attraversano il volume di ricostruzione genera misure sparse e non uniformi e quindi se analizzati con degli istogrammi molti campi di questi rimarrebbero vuoti, mostrando il vantaggio della procedura List-Mode.

Un $M$-esimo elemento di dati $s$, consiste nei dati $\{s_i; i = 1, \ldots, M\}$ dove ogni $s_i$ è dato dalla deviazione dell'angolo $\Delta\phi_i$ generato dal muone lungo l'$i$-esima traccia. La distribuzione statistica di $s_i$ è data dalla probabilità Gaussiana con funzione di densità:

$$P_i = P\left(s_i|\sigma_i\right) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-fracs_i^2(2\sigma_i^2)} \tag{3.1}$$

con la varianza $\sigma_i^2$ che è connessa deterministicamente alla $\lambda_0$ dall'integrale:

$$\sigma_i^2 = \frac{C}{p_i^2} \int_{path\ i} \lambda_0\left[r_i(l)\right] dl. \tag{3.2}$$

L'integrale va calcolato lungo l'$i$-esimo percorso dato dalla rappresentazione parametrica $r_i(l)$.

Se i dati raccolti consistono in $M$ tracce, ci sono $M$ integrali, che dopo una discretizzazione permettono di scrivere il proiettore $L$ come una matrice $M$x$N$ che connette il set di $N$ elementi $\lambda$ con il set di $M$ elementi $\sigma^2$ come $\sigma^2 = L \cdot \lambda$ o esplicitamente:

$$\sigma_i^2(\lambda) = (L \cdot \lambda)_i = \sum_{j=1}^{N} L_{ij}\lambda_j \left(i = 1, \ldots, M\right). \tag{3.3}$$

Gli errori sperimentali sono aggiunti in forma quadratica alla misura della variazione d'angolo come mostrato nella seguente equazione:

$$\sigma_i^2(\lambda) = (L \cdot \lambda)_i = \sum_{j=1}^{N} L_{ij}\lambda_j + \varepsilon_i^2 \, (i = 1, \ldots, M) \, . \tag{3.4}$$

Ogni $L_{ij}$ è valutato come il prodotto di $C/p_i^2$ volte il percorso dell'$i$-esimo percorso attraverso il $j$-esimo voxel. Per i voxel non attraversati dalla traiettoria questo valore è posto a zero.

Il problema di individuare per un data set $s$ il valore più probabile di $\lambda$ può essere risolto con una strategia del Maximum Likelihood Expetectation Maximization (MLEM)[9][10]. Nel nostro caso si riduce ad un problema di ottimizzazione che consiste nel trovare il minimo di questa funzione di costo:

$$\Psi(\lambda) = \sum_{i=1}^{M} \left[ \frac{s_i^2}{\sigma_i^2(\lambda)} + \ln \sigma_i^2(\lambda) \right] = \sum_{i=1}^{M} \left\{ \frac{s_i^2}{(L \cdot \lambda)_i + \varepsilon_i^2} + \ln \left[ (L \cdot \lambda)_i + \varepsilon_i^2 \right] \right\} \tag{3.5}$$

il gradiente $\nabla\Psi$ può essere facilmente valutato. Per il j-esimo componente troviamo:

$$(\nabla\Psi)_j = \sum_{i=1}^{M} L_{ij} \frac{(L \cdot \lambda)_i + \varepsilon_i^2 - s_i^2}{\left[ (L \cdot \lambda)_i + \varepsilon_i^2 \right]^2}. \tag{3.6}$$

Le iterazioni cominciano da un valore uniforme $\lambda^0$ che è la densità di scattering dell'aria. Poi per sviluppare l'algoritmo si deve trovare il valore di $\alpha^*$ che minimizza l'espressione:

$$\Psi \left( \lambda^0 - \alpha \nabla\Psi \right) . \tag{3.7}$$

Quindi $\lambda$ è aggiornato in accordo a:

$$\lambda^1 = \lambda^0 - \alpha^* \nabla\Psi \tag{3.8}$$

e l'iterazione successiva è calcolata cercando il nuovo gradiente $\nabla\Psi$ e il nuovo $\alpha^*$.

## 3.4 Risultati

Dai risultati degli esperimenti effettuati per convalidare la teoria si possono trarre diverse conclusioni.

### 3.4.1 Capacità di ricostruzione delle immagini

Per verificare la capacità di ricostruzione tridimensionale posizioniamo una coppia di 2 blocchi, uno di piombo e uno di acciaio e grazie ad una struttura una coppia del tutto identica sopra la precedente di altri 65 centimetri come in figura 3.1.

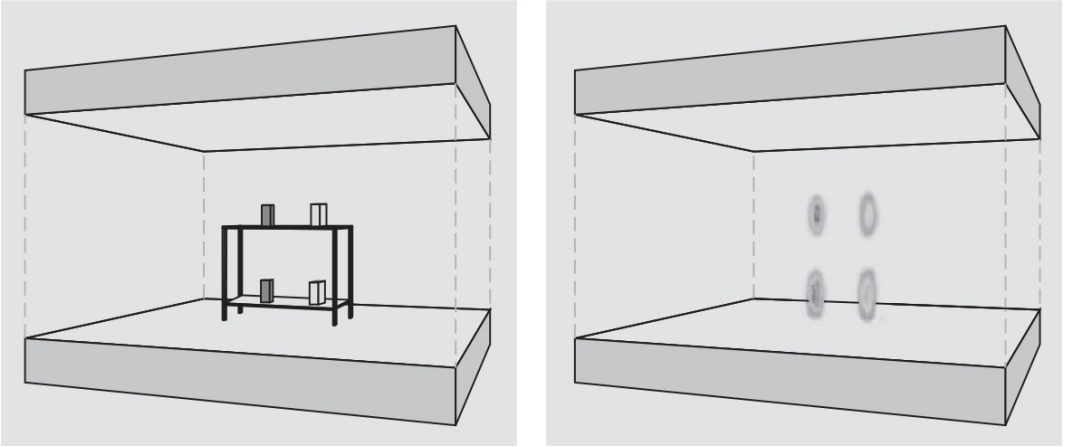

Figura 3.1: Disposizione di test.

Dai test effettuati si può subito notare che il sistema presenta le seguenti caratteristiche:

1. É in grado di riprodurre esattamente la posizione dei blocchi nello spazio del volume ispezionato.

2. L'effetto della ricostruzione con una risoluzione finita dei materiali è evidente soprattutto in verticale.

3. La densità di scattering ricostruita per i blocchi di piombo è più larga che per i blocchi di acciaio.

### 3.4.2 Identificazione dei materiali

Per verificare questo tipo di riconoscimento si è allestito un esperimento con sei blocchi di materiali e dimensioni diversi, per la precisione: rame, piombo, ferro, alluminio, tungsteno, zinco. La ricostruzione effettuata dal sistema (figura 3.2) mostra che c'è solo una lieve differenza di densità di scattering tra piombo e tungsteno.

Vista la differenza di dimensione dei blocchi i valori vanno normalizzati per avere una diretta possibilità di confronto con gli altri materiali.

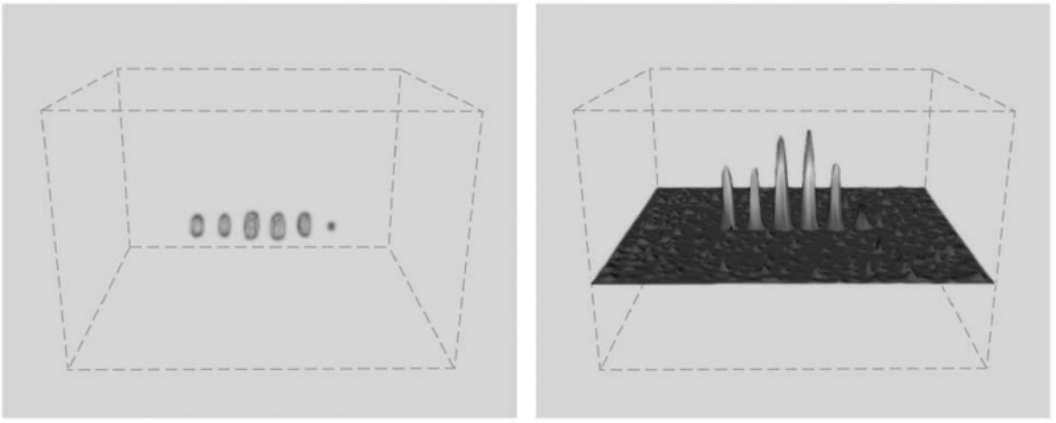

Figura 3.2: Risultato del test.

L'errore rilevato sulla misurazione della densità di scattering è del 2,8%. Praticamente si nota un errore in difetto che aumenta con l'aumentare della densità dei materiali, ma l'errore cresce in modo non lineare. Questo complica il riconoscimento e la discriminazioni tra i materiali ad alta densità. Per controllare aumentiamo la dimensione dei blocchi dei materiali e dai risultati si possono vedere che l'errore diminuisce nei materiali più densi e che quindi si riprende un comportamento più lineare, comunque la non linearità rimane troppo elevata per essere un semplice errore sperimentale e la spiegazione è dovuta all'assorbimento di muoni a bassa energia. Infatti un muone con poca energia ha la possibilità di essere assorbito e l'eventualità di tale evento dipende dalla natura del materiale e dalla sua dimensione. Quindi lo spettro di energia dei muoni utile per la misurazione dei materiali con densità più alte deve essere più alto. Comunque va tenuta anche in considerazione la posizione geografica di dove si tiene l'esperimento, in quanto i muoni possono venire influenzati anche dall'altezza sul livello del mare e dalla latitudine geomagnetica[11].

## 3.5 Conclusioni

Si può quindi concludere che uno dei più grandi vantaggi di questo tipo di tomografia è la capacità di analizzare materiali di elevato spessore grazie alla forza di

penetrazione dei muoni. Inoltre questo sistema permette di avere un'immagine tridimensionale e di classificare velocemente i tipi di materiale coinvolti nella tomografia. Si ha anche la possibilità di ricostruire volumi dalle dimensioni elevate. Infine questo tipo di tomografia non genera alcun tipo di radiazione nociva non utilizzando nessun generatore attivo di particelle ma sfruttando passivamente quelle generate dai raggi cosmici.

Capitolo 4

# Obiettivi della tesi

Lo scopo della tesi in questione è quello di occuparsi del codice che effettua la ricostruzione a partire dai dati raccolti dal prototipo costruito, infatti al fine di ricostruire le immagini tridimensionali è stato sviluppato del codice C++. Questo codice nonostante sia funzionante ed ottenga i risultati corretti è da considerare ancora in fase di sviluppo in quanto impiega un tempo considerato eccessivamente lungo per completare le operazioni affinchè sia utilizzabile per un'applicazione reale.

L'obiettivo in questo caso è quindi di rendere il codice di ricostruzione delle immagini più efficiente senza compromettere il risultato ottenuto. Le possibilità di arrivare ad un miglioramento è ottenibile concettualmente seguendo quattro percorsi:

- Mantenimento dello stesso algoritmo ma ottimizzazione delle istruzioni effettuate;
- Variazione delle strutture dati utilizzate;
- Variazioni di esecuzione dell'algoritmo;
- Creazione di un nuovo algoritmo per la ricostruzione.

Per la tesi in questione si è deciso di prendere in considerazione soltanto le prime tre possibilità, questo perchè l'algoritmo sviluppato non è ancora arrivato al suo limite e ci sono tutte le potenzialità per migliorare la sua implementazione. Inoltre la progettazione di un nuovo algoritmo comporterebbe un diverso approccio al problema

dal punto di vista scientifico quindi potrebbero non esserci tutte le competenze sulle quali lavorare ed esulerebbe dal percorso di studi.

Passiamo ad esaminare gli altri tre percorsi. Nel primo, lo scopo è quello di individuare istruzioni che vengono svolte in modo poco efficiente, per esempio ripetendo più volte gli stessi calcoli quando non necessari, quindi questo tipo di approccio non va a modificare in nessun modo nessun concetto o sviluppo dell'algoritmo. Inoltre questo tipo di ottimizzazione non va ad intaccare il risultato che rimarrebbe il medesimo.

Nel secondo tipo di approccio vengono analizzate le strutture che sono utilizzate per l'esecuzione del programma, come per esempio i tipi di dati e le strutture più o meno complesse nelle quali sono immagazzinati. Il tipo di modifiche che vengono effettuate con questo approccio hanno lo scopo di migliorare le inefficienze nell'uso delle risorse, per esempio con strutture dati dalle dimensioni più contenute o strutture poco efficienti dal punto di vista del calcolo e reperimento dei dati. Anche questa tipologia di modifiche non va a modificare lo sviluppo dell'algoritmo, che seguirebbe lo stesso percorso, né va a modificare i risultati ottenuti.

L'ultimo percorso preso in esame è il più complesso e consiste nell'analisi dell'algoritmo ed individuazioni di alcune inefficienze dovute all'ordine di esecuzione delle operazioni o effettuare delle approssimazioni nei calcoli, questa modifica va ad altereare leggermente l'algoritmo dal punto di vista pratico ma non lo modifica dal punto di vista concettuale. I risultati che si ottengono sono simili ai precedenti e se cambiano questi sono dovute all'approssimazione del calcolo da parte della macchina o scelte effettuate a priori nell'implementazione.

Nel prossimo capitolo andiamo a spiegare come si svoglerà il lavoro a partire da queste considerazioni.

Capitolo 5

# Pianificazione del lavoro

Per un buon approccio il lavoro si dividerà in fasi successive con lo scopo di aumentare progressivamente le competenze fino a padroneggiare completamente la situazione e poter svolgere il lavoro di ottimizzazione in modo adeguato. Andiamo quindi ad analizzare il lavoro delle singole fasi.

## 5.1 Studio del codice presente

Acquisite le necessarie competenze dell'argomento e del linguaggio di programmazione utilizzato, si è passato ad uno studio del codice presente al fine di capire il funzionamento, verificare come è stato implementato l'algoritmo pensato a livello pratico ed individuarne limiti.

## 5.2 Creazione di una documentazione del codice

Una volta studiato il codice originario si è stesa una documentazione ad utilizzo personale con lo scopo di aiutare la memorizzazione di quanto appreso e di capirne al meglio il suo funzionamento, inoltre questo ha facilitato la messa in opera delle modifiche in quanto permetteva durante il lavoro di individuare rapidamente le funzioni coinvolte nel processo.

## 5.3 Analisi del codice originario

Completata la fase precedente si è fatto eseguire il codice originario su un pc personale e su un una macchina multiprocessore del dipartimento di fisica dell'Università di Padova. L'operatività su una macchina personale ha lo scopo di slegare lo sviluppo dalla macchina del dipartimento al fine di non appesantirla e di velocizzare quindi il lavoro. Il funzionamento sulla macchina del dipartimento ha lo scopo di poter effettuare delle esecuzioni che costituiscano dei riferimenti prestazionali precisi, dai quali partire per misurare le prestazioni ottenute con la successiva ottimizzazione del codice, visto che la macchina dove opererà la versione definitiva sfrutta la stessa architettura.

## 5.4 Scelta delle modifiche da effettuare

Con l'analisi delle prestazioni precedenti si individuano i punti più critici del codice con i quali si spera di ottenere la maggior quantità di speedup, per esempio individuando dove viene speso più tempo dal processore, infatti non presenta alcun vantaggio concentrarsi su parti di codice che occupano una frazione limitata del tempo totale. Anche ottenendo un grande miglioramento il suo impatto sul tempo totale sarebbe minimo, mentre ottimizzando anche di poco una parte di codice o funzione che occupa gran parte del tempo di esecuzione si ha un grande impatto globale. Quindi la scelta delle modifiche è un processo da non sottovalutare per non disperdere tempo e risorse.

## 5.5 Processo di modifica

Il processo di introduzione delle modifiche è la parte che va a produrre i risultati tangibili che si sono preventivati precedentemente, è un processo che si ripete in modo ciclico per ognuna delle modifiche che si va ad introdurre, infatti appena viene completata una modifica si deve analizzare il suo impatto sul codice e verificare che non introduca variazioni del risultato o se le introduce che siano modifiche non rilevanti ai fini della sua bontà. Una volta introdotta e validata una modifica si passa alla successiva modifica. Questo processo si può suddividere in due sottofasi.

### 5.5.1 Modifica del codice

Viene modificato materialmente il codice scritto in precedenza con lo scopo di ottimizzarlo, una volta modificato il codice deve essere funzionante.

### 5.5.2 Analisi prestazionale del codice modificato

Una volta ottenuto un codice funzionante si verifica attraverso dei test prestazionali il miglioramento rispetto al passo precedente dell'ottimizzazione. Ovviamente vanno verificati con appositi strumenti anche l'integrità del risultato ottenuto.

## 5.6 Commento al lavoro

Finito il processo ciclico di modifiche vanno analizzati i risultati ottenuti nel loro complesso e valutati nel contesto di introduzione e del lavoro fatto.

Capitolo 6

# Presentazione del codice attuale

## 6.1 Struttura del codice

Il codice presente all'inizio della tesi era composto da un main, cinque interfacce e cinque classi C++.
La struttura del codice era la seguente:

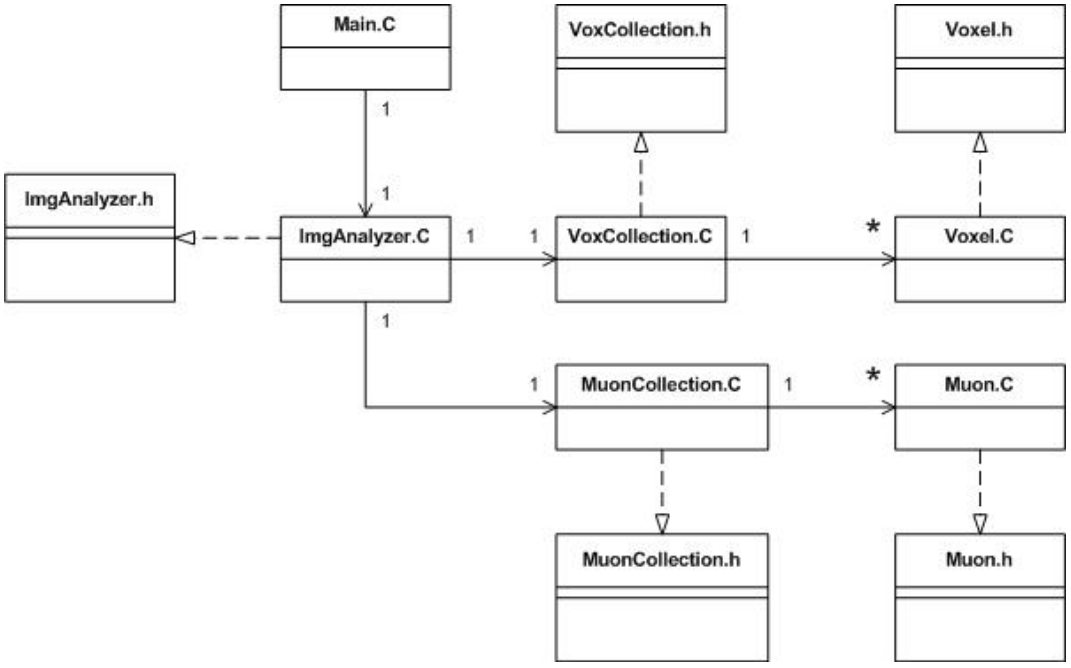


Figura 6.1: Schema delle classi.

## 6.2 Funzionamento del software

Descriviamo ora in modo discorsivo il ruolo delle classi presenti nel programma al fine di renderne chiaro il funzionamento.

La classe main svolge la funzione di interfaccia utente, controlla i parametri di ingresso che vengono dati al programma per poi creare l'oggetto ImgAnalyzer che effettua effettivamente la ricostruzione tomografica.

All'interno di ImgAnalyzer si sviluppa il cuore del software, infatti al suo interno dopo essere stato richiamato dal main vengono inizializzati tutti i parametri necessari, presi i dati in ingresso e successivamente effettuata la vera ricostruzione.

Nell'inizializzazione si creano le strutture dati di appoggio come la collezione dei muoni, quella dei voxel, attraverso rispettivamente MuonCollection e VoxCollection, e tutti i parametri fisici necessari. Le strutture dati di MuonCollection e VoxCollection sono create con al loro interno già tutta la memoria allocata per gli oggetti necessari all'elaborazione, che nel primo caso è costituito da un vettore di oggetti della classe Muon mentre nel secondo da un vettore di oggetti Voxel. Le classi che operano come struttura dati offrono tutte le funzionalità necessarie ad operare nella collezione. Le classi che invece costituiscono i singoli oggetti forniscono le funzioni per accedere a tutti i parametri al loro interno e anche le funzioni di calcolo per aggiornare i parametri al loro interno.

Nella lettura si legge dal file d'ingresso i dati dei muoni che vanno a popolare la MuonCollection andando così a riempire con dati consistenti i singoli oggetti Muon creati in precedenza.

Nella fase di ricostruzione si vanno ad analizzare i dati dei muoni, memorizzati all'interno della MuonCollection, al fine di determinare la traiettoria di ognuno di essi e quindi ricavare di conseguenza la densità media di ogni Voxel presente in VoxCollection che rappresenta il volume analizzato. Sempre in questa fase una volta determinata la densità di ogni Voxel si salva il tutto in file che rappresentano aree tridimensionali.

## 6.3 Tipologia di misure

Questo codice essendo la base di partenza del lavoro di tesi costituirà il riferimento per tutte le misurazioni dei miglioramenti al codice che andremo a introdurre in

seguito.

Ogni esecuzione del programma necessita in input un file che contiene le posizioni di ingresso e di uscita dei muoni all'interno della struttura di misurazione, nel nostro caso verranno utilizzati tre file di input, tutti contenenti le posizioni di ingresso ed uscita di venti milioni di muoni. I file in questione hanno un codice identificativo della misurazione che utilizzeremo come riferimento ai singoli campioni, i codici sono 828, 829 e 833.

### 6.3.1 Primo tipo di campione

Le misure di riferimento avranno come campione principale i seguenti parametri di Test:

- 1 milione di muoni utilizzati per la ricostruzione;

- Una struttura di Voxel con le seguenti dimensioni: 44 nell'asse $X$, 34 in $Y$ e 22 in $Z$, tutti di forma cubica di 7 cm di lato al fine di avere un giusto trade-off tra il numero di muoni incidenti per ogni voxel e la possibilità di identificare anche oggetti sufficientemente piccoli.

Per ogni file di input verranno prese le misure dei tempi di dieci esecuzioni del programma per ottenere una valutazione media in modo da evitare che lo stato momentaneo della macchina dove avviene l'esecuzione influenzi la misura. La media poi verrà considerata il tempo di riferimento per misurare il miglioramento delle singole modifiche che il codice ottiene.
Presentiamo ora in forma tabulare i risultati ottenuti e il calcolo delle relative medie, che costituiranno il riferimento per valutare gli speed up, dei tre file di ingresso.

Tabella 6.1: File di input 828 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| 1 | 0,01 | 39,9 | 2948,6 | 2988,5 |
| 2 | 0,01 | 33,8 | 2837,8 | 2871,6 |
| 3 | 0 | 33,6 | 2786,2 | 2819,8 |
| 4 | 0 | 34,6 | 2772,1 | 2806,7 |
| 5 | 0 | 33,5 | 2865,7 | 2899,2 |
| 6 | 0,01 | 33,5 | 2854,5 | 2888,0 |
| 7 | 0 | 33,5 | 2839,5 | 2873,0 |
| 8 | 0,01 | 40,3 | 2939,5 | 2979,9 |
| 9 | 0 | 34,1 | 3167,9 | 3201,9 |
| 10 | 0 | 33,0 | 2755,1 | 2788,0 |
| media | 0,004 | 35,0 | 2876,7 | 2911,7 |

Tabella 6.2: File di input 829 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| 1 | 0,01 | 39,5 | 2767,5 | 2807,0 |
| 2 | 0 | 33,6 | 2777,6 | 2811,1 |
| 3 | 0 | 33,6 | 2756,8 | 2790,4 |
| 4 | 0 | 33,5 | 2812,6 | 2846,1 |
| 5 | 0 | 33,5 | 2783,8 | 2817,3 |
| 6 | 0 | 33,5 | 2786,1 | 2819,6 |
| 7 | 0,01 | 33,5 | 2766,3 | 2799,9 |
| 8 | 0 | 34,4 | 2884,9 | 2919,2 |
| 9 | 0,02 | 35,9 | 2937,5 | 2973,4 |
| 10 | 0 | 33,1 | 2793,9 | 2826,9 |
| media | 0,004 | 34,4 | 2806,7 | 2841,1 |

Tabella 6.3: File di input 833 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| 1 | 0,01 | 38,8 | 2932,5 | 2971,3 |
| 2 | 0 | 39,4 | 2763,2 | 2802,6 |
| 3 | 0 | 33,1 | 2799,4 | 2832,5 |
| 4 | 0 | 33,1 | 2832,5 | 2865,7 |
| 5 | 0,01 | 33,8 | 2770,2 | 2803,9 |
| 6 | 0 | 32,9 | 2844,1 | 2877,0 |
| 7 | 0 | 32,9 | 2819,0 | 2851,9 |
| 8 | 0 | 32,9 | 2740,1 | 2773,0 |
| 9 | 0,01 | 32,9 | 2857,6 | 2890,5 |
| 10 | 0 | 33,0 | 2755,1 | 2788,0 |
| media | 0,003 | 34,3 | 2811,4 | 2845,6 |

Dai risultati esposti si può notare che il programma nei tre casi impiega un tempo che va dai 47 minuti ad un massimo di 53 e la media si aggira sui 48 minuti. La maggior parte del tempo di esecuzione è occupato dal tempo di ricostruzione, mentre il tempo di inizializzazione è trascurabile ai fini delle prestazioni, spesso risulta praticamente non rilevabile, e quello di lettura dei dati incide per poco più del 10%. Si può quindi considerare come riferimento il solo tempo di ricostruzione.

### 6.3.2 Secondo tipo di campione

Per una misura più completa calcoleremo il tempo di esecuzione anche con cinque milioni di muoni per il file di input 828 al fine di considerare lo speedup ottenuto anche con un grande quantitativo di muoni, situazione che si avvicina molto alla applicazione reale del codice.

Mostriamo ora i risultati ottenuti dal codice base in forma tabulare.

Tabella 6.4: File di input 828 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| 1 | 0 | 191,8 | 14661,1 | 14852,9 |
| 2 | 0 | 185,1 | 13636,0 | 13821,1 |
| 3 | 0 | 198,8 | 13703,5 | 13902,3 |
| 4 | 0 | 194,8 | 13747,0 | 13941,8 |
| media | 0 | 192,6 | 13936,9 | 14129,5 |

Il tempo di esecuzione in questo caso varia dai 230 minuti (3,8 ore) ad un massimo di 247 minuti (4,1 ore). Anche qui si confermano come nel caso precedente la trascurabilità del tempo di inizializzazione e la poca rilevanza globale del tempo di lettura dei dati in ingresso.

## 6.4 Utilizzo della memoria RAM

Al fine di valutare la qualità delle ottimizzazioni dobbiamo anche monitorare l'utilizzo della memoria di sistema, infatti ottenere un'ottimizzazione del sofwtare che va ad occupare una maggiore quantità di memoria potrebbe essere controproducente. Analizzando un grande quantitativo di dati, maggiore alle misure standard, potremmo occupare l'intera memoria di sistema e costringere il programma ad effettuare lo swap della memoria RAM su disco andando a rallentare in modo drastico le prestazioni totali e quindi non ottenere una vera ottimizzazione. Qualora si ottenesse un miglioramento che crea questo problema andrà analizzato con la dovuta cautela se l'utilizzo maggiore di risorse possa essere accettabile o meno. Vediamo ora quanta memoria occupa un'esecuzione che prevede come parametri di ingresso un milione di muoni.

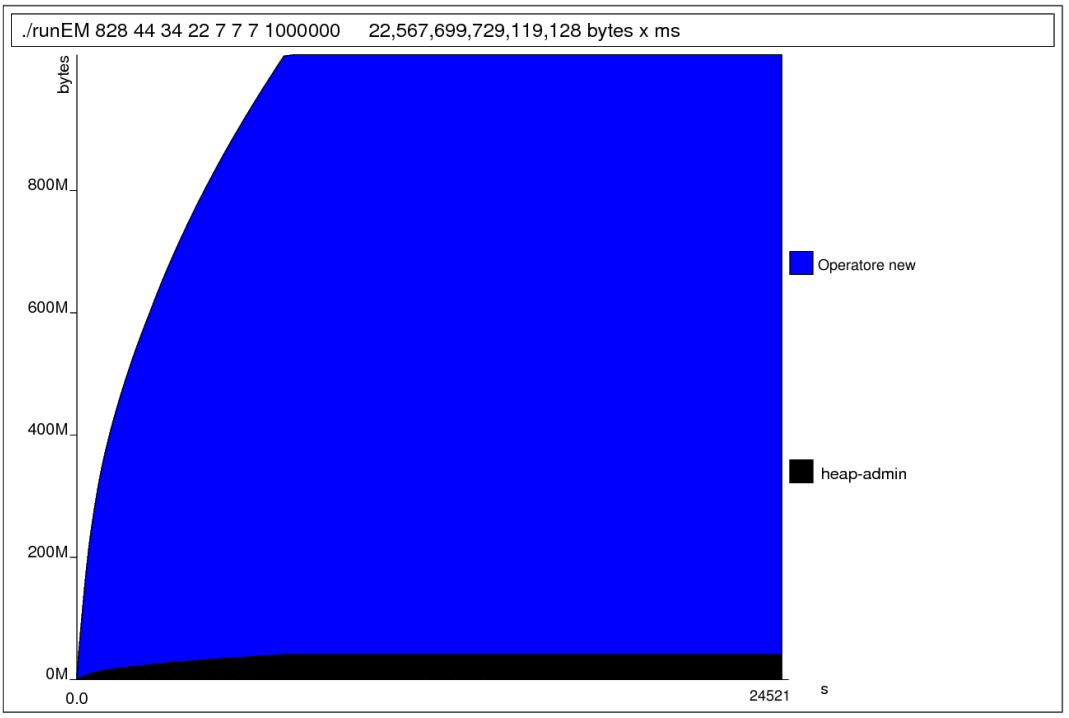


Figura 6.2: Utilizzo della Ram nel tempo.

Il tempo di esecuzione è estremamente sproporzionato rispetto alla sua durata normale, in quanto per il monitoraggio della memoria e la generazione del grafico l'uso delle risorse non è ottimale, quindi non ha alcuna rilevanza, infatti, il profiler nelle sue specifiche dichiara un tempo d'esecuzione di almeno venti volte superiore [17]. L'occupazione di memoria possiamo vedere che cresce rapidamente per poi rimanere stabile fino alla fine dell'esecuzione del programma, questo è dovuto all'inizializzazione, prima della ricostruzione, di tutte le strutture dati che verranno utilizzate come è stato spiegato precedentemente. Vediamo che la sua occupazione è mediamente attorno ad un GigaByte di memoria, per la precisione abbiamo un'occupazione media di 919,6 MegaByte. Per le successive ottimizzazioni che verranno implementate questo costituirà il parametro di riferimento.

## 6.5 Utilizzo del processore

Analizzare come il programma utilizza il processore e quali sono le funzioni software che lo utilizzano maggiormente permette di sapere su quali aspetti concentrarsi per ottenere miglioramenti tangibili senza sprecare tempo in ottimizzazioni che

potrebbero avere risultati marginali. Andiamo quindi a vedere in percentuale quali sono le funzioni che occupano la maggior parte del tempo di esecuzione (Figura 6.3).

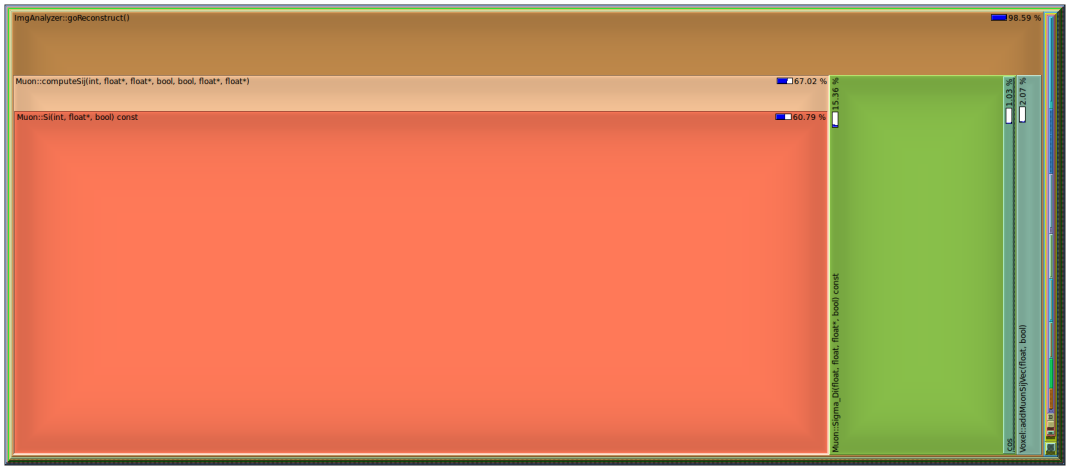


Figura 6.3: Utilizzo del processore in percentuale.

Il profiler rappresenta l'uso del processore in rettangoli. A partire dal main, il quadrato più grande, alle varie funzioni racchiuse all'interno dei rettangoli della funzione che le richiama. Si può subito notare che le funzioni che occupano la maggior parte del tempo di esecuzione è dovuto alla funzione di ricostruzione al cui interno le funzioni che hanno il maggiore peso sono ComputeSij e Si. Quindi nel prossimo capitolo saranno fatte le opportune considerazioni a proposito.

Il grafico potrebbe essere poco chiaro in quanto i rettangoli più esterni sono molto piccoli ma con un albero delle chiamate con riferimento solo alle principali funzioni risulterà più chiaro (Figura 6.4).

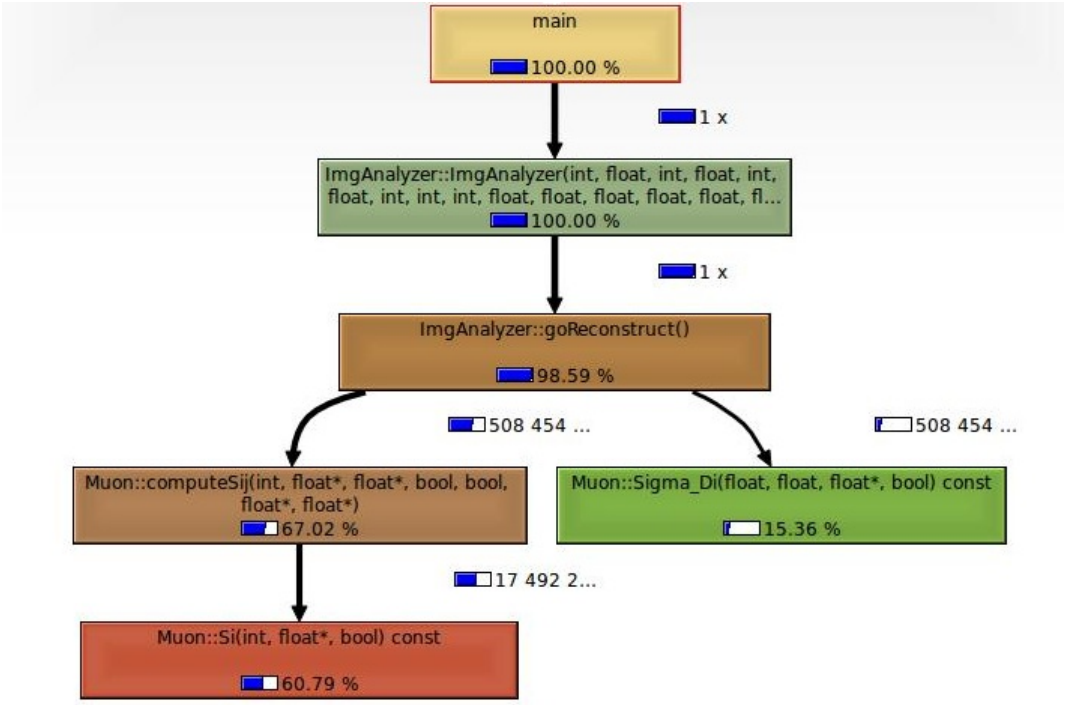


Figura 6.4: Albero delle chiamate con percentuali di utilizzo.

Capitolo 7

# Tipologia di modifiche da effettuare

Una volta individuati i punti critici del codice con l'analisi del codice originario si sono messe in piano di lavoro le seguenti modifiche:

- Sostituzione della classe Voxel con VoxelData che viene istanziata una sola volta e contenente i 5 array per i dati della vecchia classe Voxel.
- Utilizzare un puntatore di VoxelData che effettua la chiamata già all'indice corretto.
- Mettere il puntatore direttamente in VoxelCollection.
- Ottimizzazione della funzione computeSij().

Questo tipo di modifiche potrebbero portare sicuramente ad una minore leggibilità del codice da parte di chi vorrebbe modificarlo in seguito, ma in questo caso la priorità più alta spetta all'efficienza che è un punto cruciale di questo tipo di applicazione. Inoltre alcune modifiche sono molto legate tra loro, come per esempio i primi tre punti della lista, per cui verranno introdotte insieme, quindi complessivamente andranno a costituire due livelli di ottimizzazione che andremo ad analizzare nel dettaglio.

## 7.1 Primo livello di ottimizzazione

*Sostituzione della classe Voxel con VoxelData che viene istanziata una sola volta e contenente i 5 array per i dati della vecchia classe Voxel.*
*Utilizzare un puntatore di VoxelData che effettua la chiamata già all'indice corretto.*
*Mettere il puntatore direttamente in VoxelCollection.*

Il primo livello di ottimizzazione consiste nella sostituzione della classe Voxel che rappresenta un singolo voxel del sistema con una classe VoxelData che ha il compito di gestire tutti i voxel. Questo con lo scopo di elimiare il vettore che contiene tutti i Voxel all'interno della classe VoxCollection, che in realtà non sono altro che puntatori ai dati del singolo voxel, perchè implica poca località nei dati in quanto il processore deve gestire un vettore di puntatori che vanno in altre aree di memoria. Con l'eliminazione di questo vengono caricati nella cache del processore direttamente i vettori che contengono i dati, quindi è molto più probabile che si sfrutti la località del processore per l'accesso ai dati. Da questa modifica ci attendiamo di ridurre i tempi di esecuzione della parte di ricostruzione grazie appunto allo sfruttamento della località. Le altre due modifiche sono strettamente legate a questa introduzione in quanto ora ogni Voxel prima era identificato da un codice di Voxel che ora va ad identificare la sua posizione dei suoi parametri nei vettori relativi, mentre il puntatore all'oggetto VoxelData è creato all'inizio dell'esecuzione di VoxCollection e quindi sono disponibili velocemente i 5 vettori con i parametri.

## 7.2 Secondo livello di ottimizzazione

*Ottimizzazione della funzione computeSij().*

Il secondo livello di ottimizzazione consiste nell'ottimizzazione della funzione ComputeSij, situata all'interno della classe Muon. Essa infatti è la funzione più chiamata dal programma occupando circa il 68 % del tempo di esecuzione totale ma al suo interno viene chiamata più volte anche la funzione Si, sempre della classe Muon, che a sua volta occupa circa l'85 % del tempo di esecuzione di ComputeSij, pari al 58 % totale. Quindi per la maggior parte del tempo occupato dalla funzione

ComputeSij è dovuto alla funzione Si e quindi in realtà influisce nel totale solo 10 % del tempo complessivo. Si presenta quindi evidente che un miglioramento dell'efficienza di queste due funzioni porterebbe ad un grosso impatto sul tempo di esecuzione totale del programma.

Capitolo 8

# Risultati ottenuti

Le modifiche che dovremmo effettuare alla struttura del codice vengono introdotte in modo graduale e misurato il miglioramento ottenuto con la singola modifica. Per ogni livello di ottimizzazione che introdurremo, presenteremo le modifiche effettuate, il motivo per cui vengono effettuate e utilizzeremo la stessa metrica presa in esame per il codice iniziale per valutare il miglioramento ottenuto.

## 8.1 Primo livello di ottimizzazione

### 8.1.1 Modifiche introdotte

Il primo livello di ottimizzazione consiste nella sostituzione della classe Voxel che rappresenta un singolo voxel del sistema con una classe VoxelData che ha il compito di gestire tutti i voxel. Questa modifica ha lo scopo di elimiare il vettore che contiene tutti i Voxel che in realtà non sono altro che puntatori ai dati del singolo voxel, ciò implica poca località nei dati in quanto il processore deve gestire un vettore di puntatori che vanno in altre aree di memoria. Con l'eliminazione di questo vengono caricati direttamente i vettori che contengono i dati quindi è molto più probabile che si sfrutti la località del processore per l'accesso ai dati. Da questa modifica ci attendiamo di ridurre i tempi di esecuzione della parte di ricostruzione grazie appunto allo sfruttamento della località.

### 8.1.2 Risultati ottenuti

#### Primo campione

Presentiamo ora in forma tabellare i risultati delle singole esecuzioni e la loro variazione a seconda del tempo di riferimento.

Tabella 8.1: File di input 828 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Var % | Reconstruction time | Var % | Total time | Var % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 29,6 | 15,40% | 2113,6 | 26,53% | 2143,1 | 26,39% |
| 2 | 0,01 | 31,0 | 11,40% | 2224,8 | 22,66% | 2255,8 | 22,53% |
| 3 | 0 | 31,2 | 10,74% | 2187,8 | 23,95% | 2219,0 | 23,79% |
| 4 | 0 | 30,8 | 11,94% | 2196,4 | 23,65% | 2227,2 | 23,51% |
| 5 | 0 | 30,5 | 12,74% | 2200,3 | 23,51% | 2230,8 | 23,38% |
| 6 | 0 | 29,1 | 16,77% | 2247,7 | 21,86% | 2276,8 | 21,80% |
| 7 | 0 | 30,5 | 12,74% | 2196,0 | 23,66% | 2226,5 | 23,53% |
| 8 | 0 | 36,7 | -4,82% | 2208,8 | 23,22% | 2239,5 | 23,09% |
| 9 | 0 | 30,1 | 14,06% | 2186,6 | 23,99% | 2216,7 | 23,87% |
| 10 | 0,01 | 30,3 | 13,23% | 2185,8 | 24,02% | 2216,1 | 23,89% |

Tabella 8.2: File di input 829 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Var % | Reconstruction time | Var % | Total time | Var % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 28,1 | 18,41% | 2120,6 | 24,44% | 2148,7 | 24,37% |
| 2 | 0 | 29,5 | 14,29% | 2204,5 | 21,45% | 2234,0 | 21,37% |
| 3 | 0 | 30,6 | 11,09% | 2214,8 | 21,09% | 2245,4 | 20,97% |
| 4 | 0 | 30,7 | 10,68% | 2188,5 | 22,03% | 2219,2 | 21,89% |
| 5 | 0 | 29,9 | 13,12% | 2195,3 | 21,78% | 2225,2 | 21,68% |
| 6 | 0 | 31,4 | 8,88% | 2204,1 | 21,47% | 2235,5 | 21,32% |
| 7 | 0 | 29,8 | 13,41% | 2167,0 | 22,79% | 2196,8 | 22,68% |
| 8 | 0 | 34,3 | 0,40% | 2106,7 | 24,94% | 2141,0 | 24,64% |
| 9 | 0 | 33,7 | 1,97% | 2187,5 | 22,06% | 2221,3 | 21,82% |
| 10 | 0,01 | 29,3 | 14,98% | 2152,6 | 23,30% | 2181,9 | 23,20% |

Tabella 8.3: File di input 833 - Tempi in secondi

| Esecuzione | Initialization time | Reading time | Var % | Reconstruction time | Var % | Total time | Var % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 32,6 | 4,85% | 2253,2 | 19,85% | 2285,8 | 19,67% |
| 2 | 0,01 | 31,3 | 8,67% | 2211,5 | 21,34% | 2242,8 | 21,18% |
| 3 | 0 | 31,2 | 8,84% | 2230,1 | 20,68% | 2261,3 | 20,53% |
| 4 | 0 | 28,9 | 15,56% | 2172,3 | 22,73% | 2201,2 | 22,65% |
| 5 | 0 | 29,0 | 15,47% | 2165,6 | 22,97% | 2194,6 | 22,88% |
| 6 | 0 | 30,7 | 10,48% | 2072,7 | 26,27% | 2103,4 | 26,08% |
| 7 | 0,01 | 32,6 | 4,85% | 2160,0 | 23,17% | 2192,7 | 22,95% |
| 8 | 0 | 28,4 | 17,13% | 2143,5 | 23,75% | 2171,9 | 23,68% |
| 9 | 0,01 | 29,7 | 13,28% | 2145,8 | 23,67% | 2175,6 | 23,55% |
| 10 | 0 | 28,4 | 17,28% | 2142,4 | 23,80% | 2170,7 | 23,72% |

Si può osservare che il miglioramento ottenuto nel tempo di ricostruzione varia da poco meno del 20% ad un massimo di poco superiore al 26%.

Di seguito mettiamo in risalto qual'è stato il guadagno e lo speedup medio ottenuto:

Tabella 8.4: Resoconto sul File di input 828 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0,002 | 31,0 | 2194,8 | 2225,1 |
| reference | 0,004 | 35,0 | 2876,7 | 2911,7 |
| speedup | | 1,13 | 1,31 | 1,31 |
| miglioramento | | 11,42% | 23,71% | 23,58% |

Tabella 8.5: Resoconto File di input 829 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0,001 | 30,7 | 2174,2 | 2204,9 |
| reference | 0,004 | 34,4 | 2806,7 | 2841,1 |
| speedup | | 1,12 | 1,29 | 1,29 |
| miglioramento | | 10,72% | 22,54% | 22,39% |

Tabella 8.6: Resoconto sul File di input 833 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0,003 | 30,3 | 2169,7 | 2200,0 |
| reference | 0,003 | 34,3 | 2811,4 | 2845,6 |
| speedup | | 1,13 | 1,30 | 1,29 |
| miglioramento | | 11,64% | 22,82% | 22,69% |

Vediamo che in tutti e tre i casi esaminati si ha un miglioramento del tempo di ricostruzione medio del 23% ed uno speedup di circa 1,3.

Mostriamo ora graficamente qual'è il risparmio di tempo ottenuto nella fase di ricostruzione rispetto al passo precendete

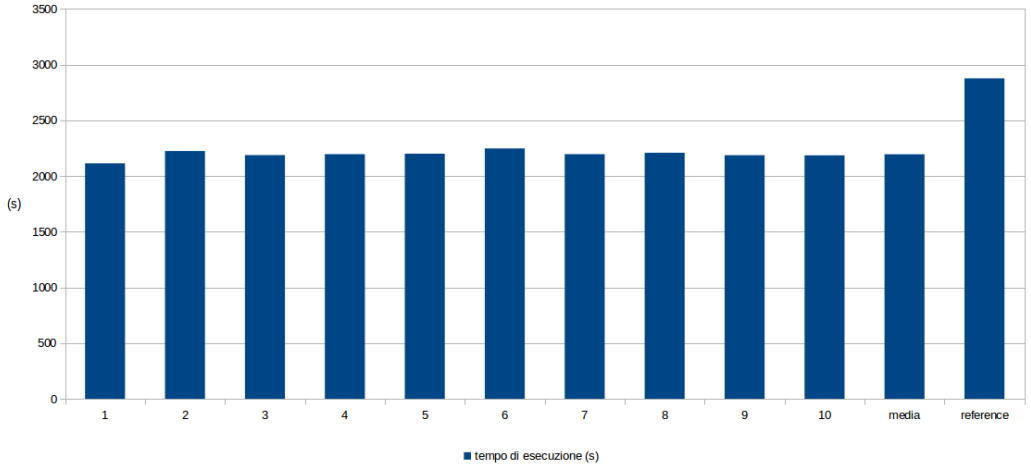


Figura 8.1: File di input 828.

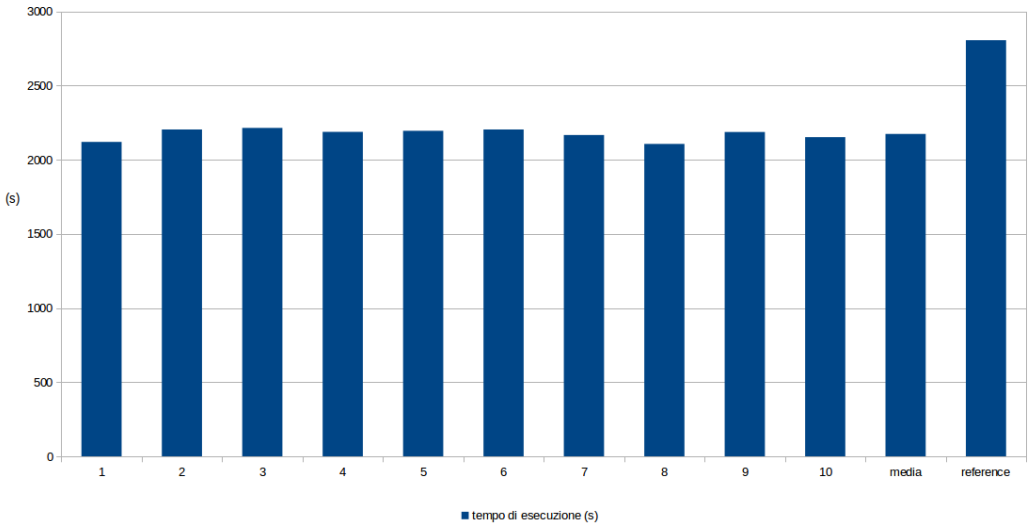


Figura 8.2: File di input 829.

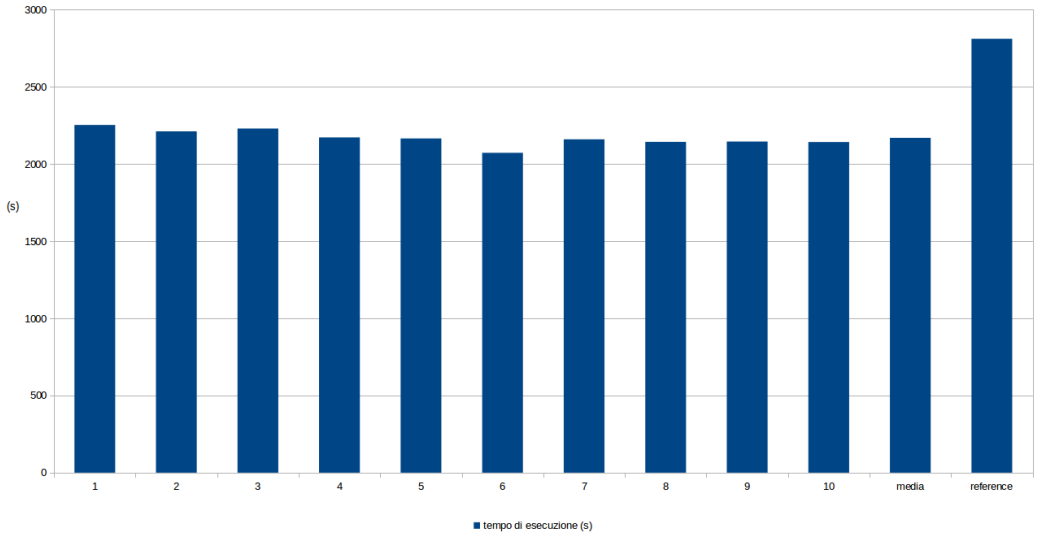


Figura 8.3: File di input 833.

Si può notare in modo evidente l'abbassamento dei tempi di esecuzione ottenuto rispetto alla versione precedente del software.

### Secondo campione

Analizziamo ora le prestazioni ottenute con il secondo tipo di campione di valutazione, al fine di verificare se mantiene i risultati ottenuti precedentemente. Presentiamo di seguito le tabelle dei risultati.

Tabella 8.7: File di input 828 - Tempi in secondi

| Esecuzione | initialization | reading | % | reconstruction | % | total | % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 176,2 | 8,54% | 10895,0 | 21,83% | 11072,0 | 21,64% |
| 2 | 0,01 | 175,9 | 8,69% | 10783,0 | 22,63% | 10959,0 | 22,44% |
| 3 | 0 | 171,0 | 11,22% | 10811,8 | 22,42% | 10982,8 | 22,27% |
| 4 | 0 | 169,8 | 11,86% | 10934,2 | 21,54% | 11104,0 | 21,41% |
| media | 0,0025 | 173,2 | | 10856,0 | | 11029,5 | |
| reference | 0 | 192,6 | | 13936,9 | | 14129,5 | |
| speedup | | 1,11 | | 1,28 | | 1,28 | |
| miglioramento | | 10,08% | | 22,11% | | 21,94% | |

Si può notare che si ottiene un miglioramento medio del 22% che conferma il risultato del primo campione e andiamo a mostrare graficamente i risultati confrontadoli con il tempo di esecuzione di riferimento.

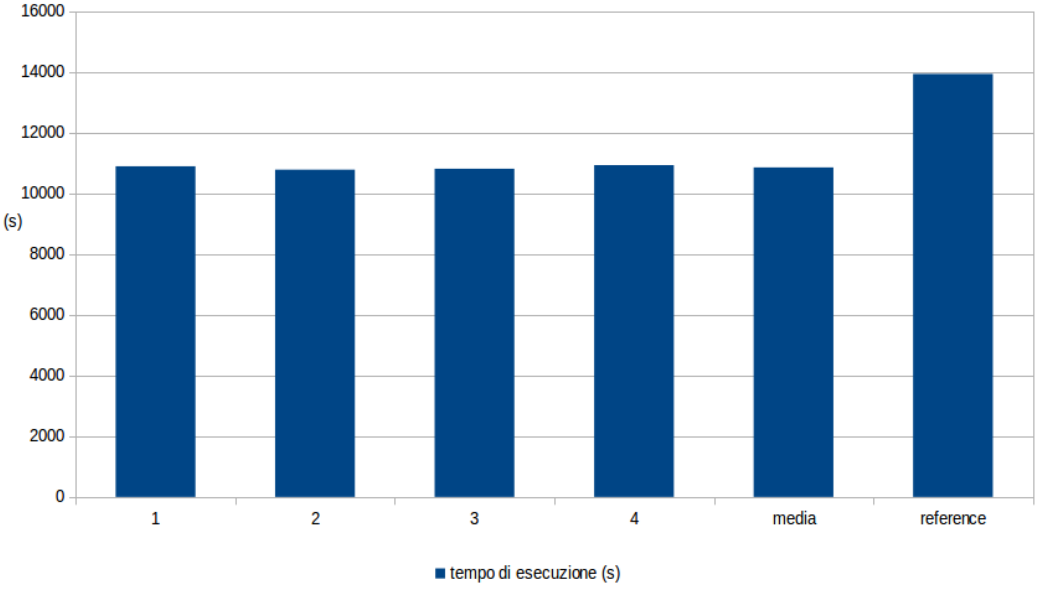


Figura 8.4: File di input 828.

Si vede anche in questo caso che l'abbassamento consistente rispetto al caso precedente.

### 8.1.3 Utilizzo della RAM

Controlliamo ora l'utilizzo della memoria RAM da parte del software dopo l'introduzione di questa ottimizzazione.

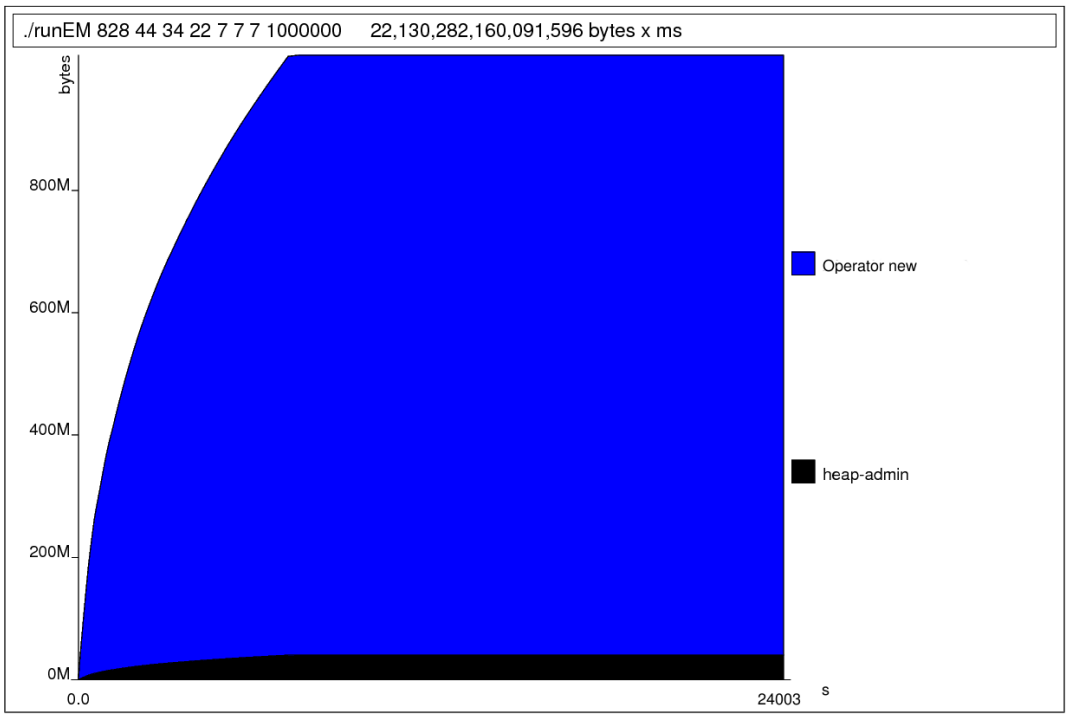


Figura 8.5: Utilizzo della Ram nel tempo.

La quantità media di RAM utilizzata è di 922 MegaByte, quindi è leggermente aumentata rispetto alla soluzione originale. Un aumento dello 0,3 % può essere considerato comunque trascurabile e dovuto alla particolare situazione della misurazione. Comunque si ottiene un aumento prestazionale mantenendo praticamente costante utilizzo delle risorse.

### 8.1.4 Commento

Questa modifica introduce un miglioramento delle prestazioni nell'ordine del 22%. Visto che questo non va ad influire sulla qualità del risultato, che è esattamente lo

stesso, possiamo considerarla una modifica ottimale e il risultato prestazionale è dovuto alla struttura di salvataggio temporanea più efficiente dal punto di vista dei tempi di accesso.

## 8.2 Secondo livello di ottimizzazione

### 8.2.1 Modifiche introdotte

Il secondo livello di ottimizzazione consiste nell'ottimizzazione della funzione ComputeSij, situata all'interno della classe Muon. Essa infatti è la funzione più chiamata dal programma occupando circa il 68% del tempo di esecuzione totale ma al suo interno viene chiamata più volte la funzione Si, sempre della classe Muon, che a sua volta occupa circa l'85 % del tempo di esecuzione di ComputeSij, pari al 58 % totale. Quindi per la maggior parte del tempo occupato dalla funzione ComputeSij è dovuto alla funzione Si e quindi in realtà influisce nel totale solo 10 % del tempo complessivo. Si presenta quindi evidente che un miglioramento dell'efficienza di queste due funzioni porterebbe ad un grosso impatto sul tempo di esecuzione totale del programma.

### 8.2.2 Risultati ottenuti

#### Primo campione

Presentiamo ora in forma tabellare i risultati delle singole esecuzioni e la loro variazione a seconda del tempo di riferimento per quello che riguarda il secondo livello di ottimizzazione del software.

Tabella 8.8: File di input 828 - Tempi in secondi

| Esecuzione | Initialization time | Reading Time | Var % | Reconstruction time | Var % | Total time | % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 33,7 | 3,73% | 1557,6 | 45,86% | 1591,2 | 45,35% |
| 2 | 0 | 27,6 | 21,09% | 1619,9 | 43,69% | 1647,5 | 43,42% |
| 3 | 0 | 29,9 | 14,60% | 1662,9 | 42,19% | 1692,8 | 41,86% |
| 4 | 0 | 27,7 | 20,81% | 1580,2 | 45,07% | 1607,9 | 44,78% |
| 5 | 0 | 27,5 | 21,32% | 1580,8 | 45,05% | 1608,3 | 44,76% |
| 6 | 0 | 27,5 | 21,26% | 1568,2 | 45,48% | 1595,8 | 45,19% |
| 7 | 0 | 27,6 | 21,06% | 1591,1 | 44,69% | 1618,7 | 44,41% |
| 8 | 0 | 27,5 | 21,24% | 1564,4 | 45,62% | 1591,9 | 45,33% |
| 9 | 0 | 27,5 | 21,24% | 1575,6 | 45,23% | 1603,2 | 44,94% |
| 10 | 0 | 27,7 | 20,78% | 1564,4 | 45,62% | 1592,1 | 45,32% |

Tabella 8.9: File di input 829 - Tempi in secondi

| Esecuzione | Initialization time | Reading Time | Var % | Reconstruction time | Var % | Total time | % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 33,7 | 2,23% | 1697,3 | 39,53% | 1731,0 | 39,07% |
| 2 | 0 | 27,7 | 19,66% | 1559,2 | 44,45% | 1586,8 | 44,15% |
| 3 | 0,01 | 27,5 | 20,04% | 1554,2 | 44,63% | 1581,7 | 44,33% |
| 4 | 0 | 27,6 | 19,81% | 1550,8 | 44,75% | 1578,4 | 44,45% |
| 5 | 0 | 27,5 | 20,10% | 1559,1 | 44,45% | 1586,6 | 44,16% |
| 6 | 0 | 27,7 | 19,55% | 1552,1 | 44,70% | 1579,8 | 44,40% |
| 7 | 0 | 27,8 | 19,20% | 1560,2 | 44,41% | 1588,0 | 44,11% |
| 8 | 0,01 | 17,6 | 48,95% | 1556,3 | 44,55% | 1583,9 | 44,25% |
| 9 | 0 | 27,8 | 19,28% | 1559,8 | 44,43% | 1587,6 | 44,12% |
| 10 | 0 | 27,7 | 19,66% | 1556,7 | 44,54% | 1584,4 | 44,23% |

Tabella 8.10: File di input 833 - Tempi in secondi

| Esecuzione | Initialization time | Reading Time | Var % | Reconstruction time | Var % | Total time | % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 30,3 | 11,56% | 1564,2 | 44,36% | 1594,2 | 43,98% |
| 2 | 0 | 27,7 | 19,17% | 1575,3 | 43,97% | 1603,0 | 43,67% |
| 3 | 0 | 27,8 | 18,79% | 1578,0 | 43,87% | 1605,8 | 43,57% |
| 4 | 0 | 27,7 | 19,32% | 1571,8 | 44,09% | 1599,4 | 43,79% |
| 5 | 0 | 27,6 | 19,44% | 1579,6 | 43,81% | 1607,2 | 43,52% |
| 6 | 0 | 27,7 | 19,26% | 1579,7 | 43,81% | 1607,3 | 43,52% |
| 7 | 0 | 27,6 | 19,41% | 1586,7 | 43,56% | 1614,3 | 43,27% |
| 8 | 0 | 27,7 | 19,17% | 1579,0 | 43,83% | 1606,7 | 43,54% |
| 9 | 0 | 27,6 | 19,55% | 1574,1 | 44,01% | 1601,6 | 43,72% |
| 10 | 0 | 27,6 | 19,44% | 1573,9 | 44,02% | 1601,5 | 43,72% |

Si può osservare che il miglioramento ottenuto nel tempo di ricostruzione varia da poco meno del 40% ad un massimo di poco inferiore al 46%.

Di seguito mettiamo in risalto qual'è stato il guadagno e lo speedup medio ottenuto:

Tabella 8.11: Resoconto sul File di input 828 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0 | 28,4 | 1586,5 | 1614,9 |
| reference | 0,004 | 35,0 | 2876,7 | 2911,7 |
| speedup | | 1,23 | 1,81 | 1,80 |
| miglioramento | | 18,71% | 44,85% | 44,54% |

Tabella 8.12: Resoconto sul File di input 829 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0,002 | 27,2 | 1570,6 | 1598,8 |
| reference | 0,004 | 34,4 | 2806,7 | 2841,1 |
| speedup | | 1,26 | 1,79 | 1,78 |
| miglioramento | | 20,85% | 44,04% | 43,73% |

Tabella 8.13: Resoconto File di input 833 - Tempi in secondi

| | Initialization time | Reading time | Reconstruction time | Total time |
|---|---|---|---|---|
| media | 0 | 27,9 | 1576,2 | 1604,1 |
| reference | 0,003 | 34,3 | 2811,4 | 2845,6 |
| speedup | | 1,23 | 1,78 | 1,77 |
| miglioramento | | 18,51% | 43,93% | 43,63% |

Vediamo che in tutti e tre i casi esaminati si ha un miglioramento del tempo di ricostruzione medio del 44% ed uno speedup di circa 1,8 rispetto al tempo originale.

Mostriamo ora graficamente qual'è il risparmio di tempo ottenuto nella fase di ricostruzione rispetto al passo precendete e al tempo di riferimento.

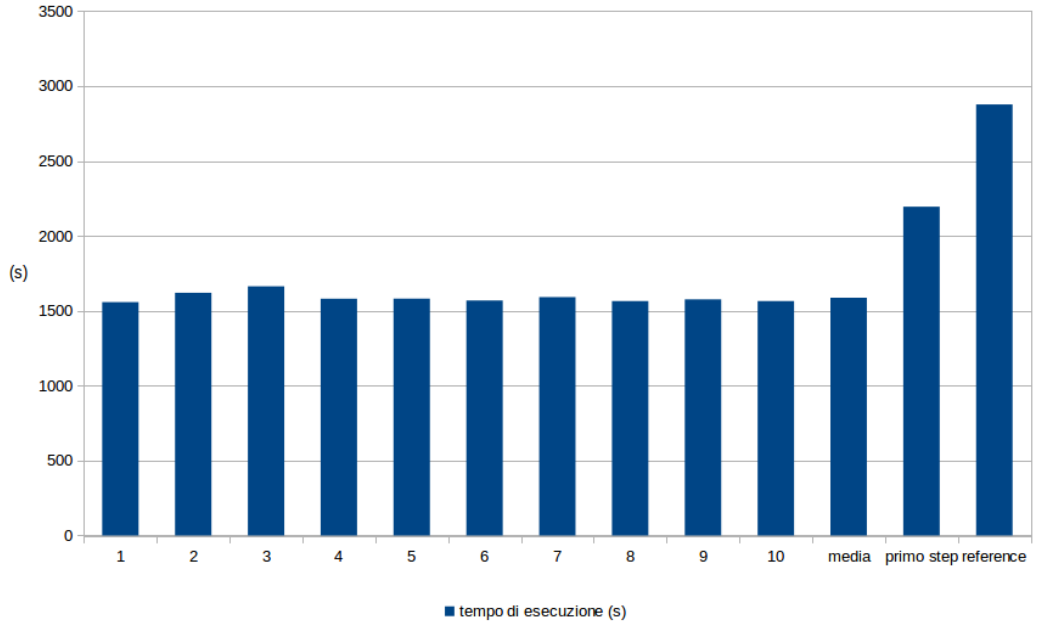


Figura 8.6: File di input 828.

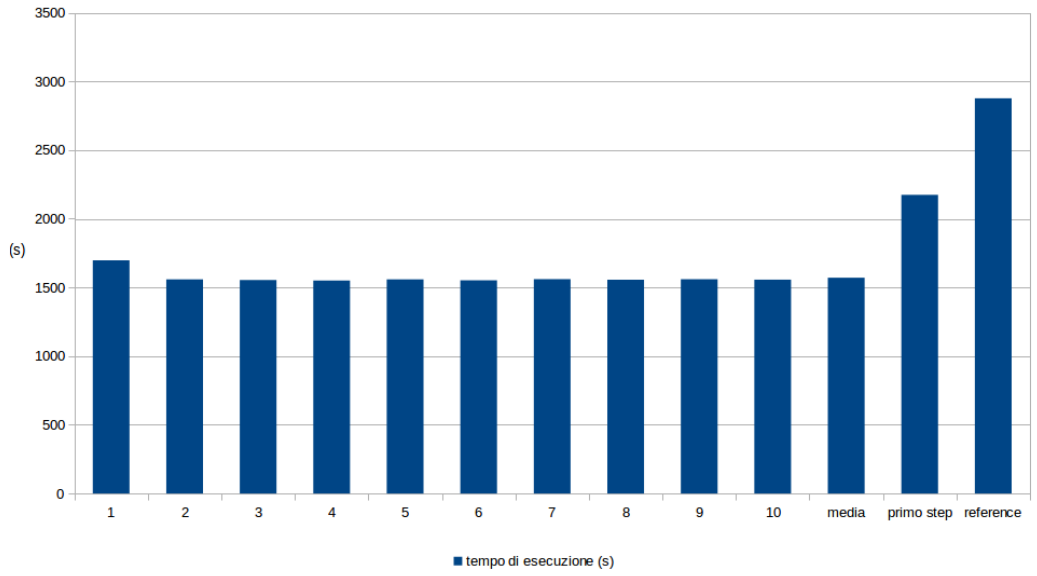


Figura 8.7: File di input 829.

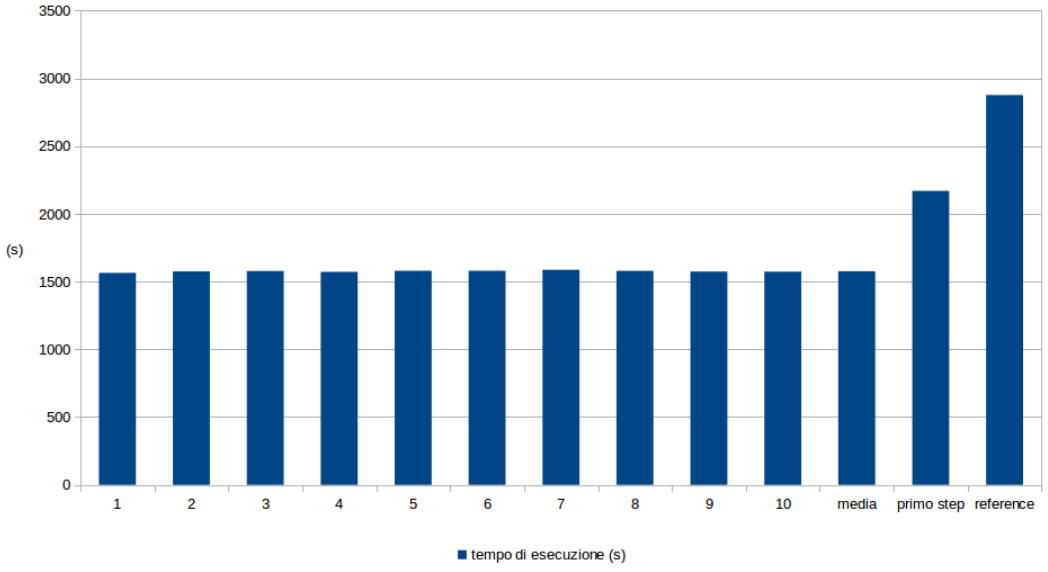


Figura 8.8: File di input 833.

Si può notare in modo evidente l'abbassamento dei tempi di esecuzione rispetto al caso precedente e quasi il dimezzamento rispetto al tempo originario.

### Secondo campione

Analizziamo ora le prestazioni ottenute con il secondo tipo di campione di valutazione, al fine di verificare se mantiene i risultati ottenuti precedentemente. Presentiamo di seguito le tabelle dei risultati.

Tabella 8.14: File di input 828 - Tempi in secondi

| Esecuzione | initialization | reading | % | reconstruction | % | total | % |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 176,9 | 8,18% | 8376,9 | 39,89% | 8553,8 | 39,46% |
| 2 | 0 | 166,7 | 13,44% | 8079,9 | 42,03% | 8246,6 | 41,64% |
| 3 | 0,01 | 174,7 | 9,32% | 8481,9 | 39,14% | 8656,6 | 38,73% |
| 4 | 0 | 169,5 | 12,03% | 8419,7 | 39,59% | 8589,2 | 39,21% |
| media | 0,0025 | 171,9 | | 8339,6 | | 8511,6 | |
| reference | 0 | 192,6 | | 13936,9 | | 14129,5 | |
| speedup | | 1,12 | | 1,67 | | 1,66 | |
| miglioramento | | 10,74% | | 40,16% | | 39,76% | |

Si può notare che si ottiene un miglioramento medio del 40% che conferma il risultato del primo campione e andiamo a mostrare graficamente i risultati confrontadoli con il tempo di esecuzione dello step di ottimizzazione precedente e quello di riferimento.

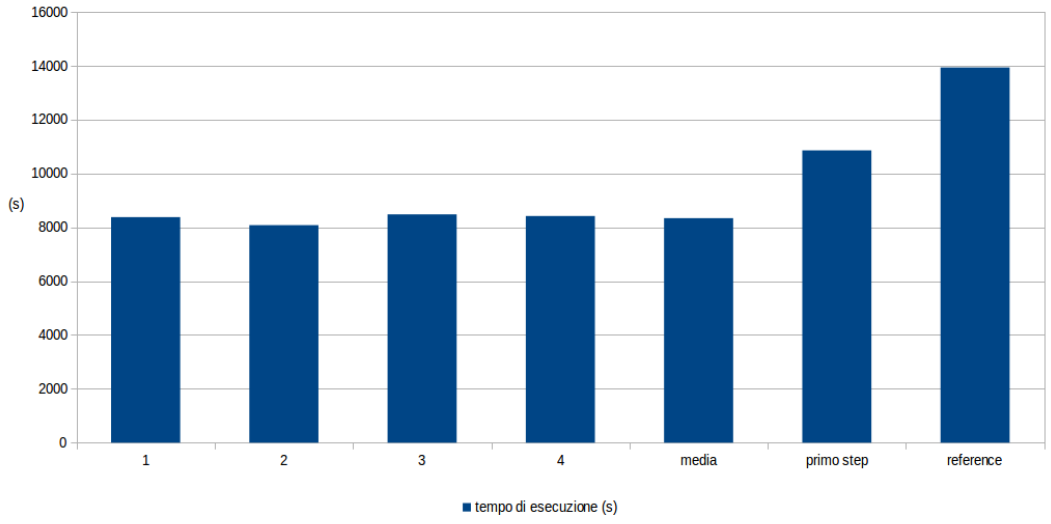


Figura 8.9: File di input 828.

Si vede immediatamente che si ottiene un sensibile miglioramento delle prestazioni soprattutto se raffrontato con il tempo di riferimento.

### 8.2.3 Utilizzo della RAM

Controlliamo ora l'utilizzo della memoria RAM da parte del software dopo l'introduzione di questa ottimizzazione per verificarne la bontà.

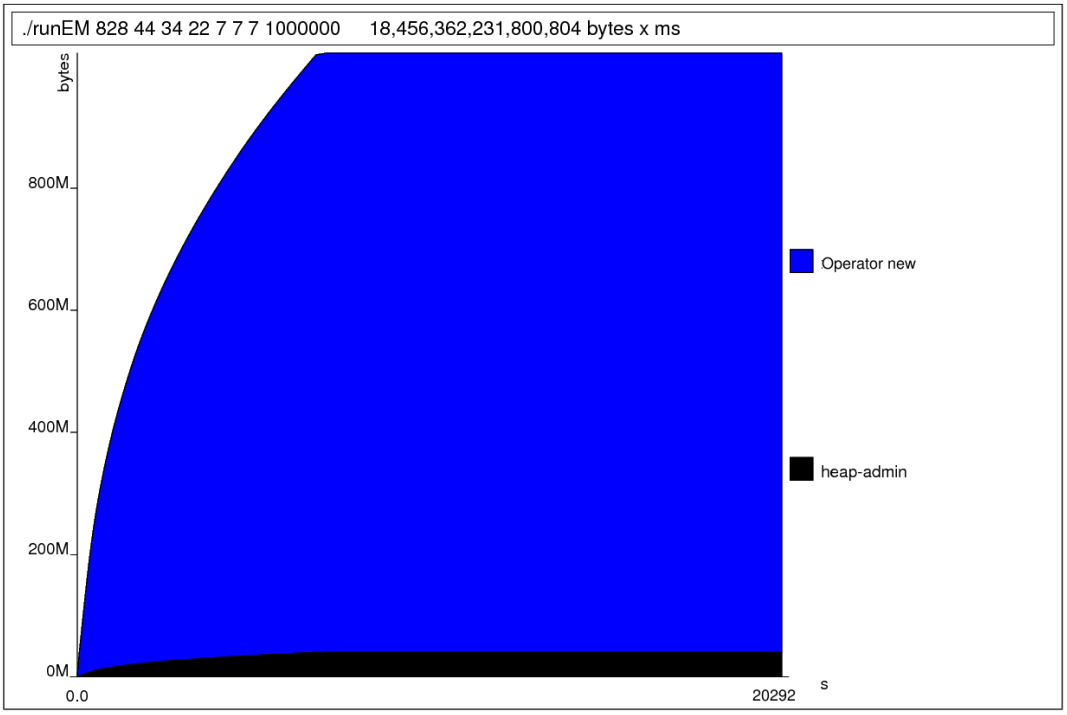


Figura 8.10: Utilizzo della Ram nel tempo.

La quantità media di RAM utilizzata è di 909,5 MegaByte, quindi è leggermente calata rispetto alla soluzione originale. Una diminuzione dell'1 % può essere considerata comunque trascurabile e dovuta alla particolare situazione della misurazione. Comunque si ottiene un aumento prestazionale mantenendo praticamente costante utilizzo delle risorse.

### 8.2.4 Commento

Questa modifica introduce un miglioramento delle prestazioni totale nell'ordine del 40% quindi circa del 20% presa singolarmente. Inoltre, come la precedente, non va

ad influire sulla qualità del risultato che è esattamente lo stesso e non utilizza una maggiore quantità di risorse, quindi possiamo considerarla come la precedente una modifica ottimale.

Capitolo 9

# Sviluppi futuri

Le ottimizzazioni presentate in questa tesi sono tutte indipendenti dall'architettura utilizzata dal sistema, queste avrebbero portato ad un incremento prestazionale indipendentemente dalla macchina sulla quale venivano utilizzate. Il percorso completo di ottimizzazione comporta anche lo studio dell'architettura dell'elaboratore dove il programma deve essere eseguito, infatti a seconda della configurazione usata ci sono delle operazioni che risultano più efficienti di altre. Ciò è dovuto esclusivamente all'hardware utilizzato, e queste istruzioni se utilizzate su dispositivi diversi potrebbero addirittura portare a dei peggioramenti prestazionali. Un successivo sviluppo quindi, se l'ottimizzazione ottenuta dal lavoro fosse ritenuta non sufficiente, comporterebbe il passaggio all'introduzione a questo tipo di modifiche. Sarebbe un lavoro che richiederebbe la ricerca della documentazione tecnica del processore, affidarsi a compilatori ottimizzati esclusivamente per l'architettura in questione ed una forte e lunga sperimentazione sulla macchina. Al contempo questo lavoro porterebbe a legare l'applicazione esclusivamente al tipo di architettura scelto quindi si può optare per questa scelta solo se si è certi di non abbandonarne l'architettura per diverso tempo.

Un ulteriore miglioramento delle prestazioni si potrebbe ottenere decidendo di approssimare il risultato effettuando moltiplicazioni e divisioni in aritmetica a virgola fissa. Questo tipo di scelta porta ad una perdita di precisione ma al contempo un aumento di prestazioni, in quanto le operazioni di questo tipo sono più veloci di quelle in virgola mobile. Questo gap prestazionale, con l'affinamento delle architetture dei processori moderni, è progressivamente diminuito ma nel nostro specifico

caso anche un minimo incremento prestazionale nel singolo ciclo potrebbe avere un impatto rilevante visto l'elevato numero di iterazioni. Questo tipo di modifica richiede uno studio molto dettagliato sui valori coinvolti al fine di non compromettere il risultato con un eccessiva perdita di precisione. Questo lavoro è studio nella tesi in questione era arrivato allo studio dei valori coinvolti nei calcoli e ovviamente costituisce un buon punto di partenza per i successivi miglioramenti.

Un'altro settore di sviluppo sarebbe la riduzione di impatto sulla memoria RAM al fine di ridurre le richieste hardware utilizzando tipi di dato meno precisi ma dal più ridotto impatto sulla memoria andando quindi a scegliere un corretto trade-off tra precisione e utilizzo delle risorse.

Capitolo 10

# Conclusioni

Lo scopo di questa tesi e del suo lavoro era quello di migliorare le prestazioni del prototipo di tomografia muonica funzionante del dipartimento di fisica dell'università di Padova senza compromettere la qualità dei risultati.

Questo obiettivo ha implicato uno studio del problema della tomografia muonica dal punto divista fisico, questo al fine di poter comprendere dal codice originario tutti i dettagli del suo funzionamento. Poi è necessitato un'analisi dal punto di vista prestazionale del software con il supporto di strumenti evoluti di profiling. Questo lavoro ha permesso di ottenere informazioni molto dettagliate su quali erano le lacune prestazionali del programma con la possibilità di indirizzare il lavoro in modo molto preciso.

Successivamente sono state pianificate, a partire dai dati ottenuti nel precedente studio, tutte le modifiche necessarie per ottenere un miglioramento prestazionale significativo.

Le ottimizzazioni che si sono implementate hanno portato risultati tangibili arrivando quasi al dimezzamento del tempo d'esecuzione senza utilizzare un maggior numero di risorse né compromettere in nessun modo il risultato che non è stato approssimato.

Inoltre nell'ultimo periodo di tesi si è lavorato sugli sviluppi futuri fornendo un buon punto di partenza per quel che riguarda un ulteriore speed up del codice utilizzando un'approssimazione utilizzando aritmetica fixed point.

Si può quindi considerare l'intero sviluppo decisamente positivo in quanto si è fatto un consistente passo avanti nelle prestazioni del codice rendendo quindi più

vicina un'applicazione pratica e reale del progetto. Inoltre si sono individuati i successivi percorsi di sviluppo sui quali si può sviluppare ulteriormente l'ottimizzazione del progetto.

Inoltre si sono acquisite competenze molto utili come per esempio l'utilizzo di tool di profiling e ottimizzazione di codice C++.

# Bibliografia


[1] L.W. Alvarez et. al., *Science Magazine*, p. 832-839, 1970

[2] S. Buontempo et. al., *Muon radiography of volcanoes and the challenge at Mt. Vesuvius*, 2008

[3] G. Z. Moliere, Z. Naturforsch 2a e 3a, p.133, 1947 e 1948

[4] H.A. Bethe, Phys. Rev. 89, p.1256, 1953

[5] K. Borozdin et. al., *Radiographic imaging with cosmic-ray muons*, Nature vol. 422, p. 277, 2003

[6] W. Priedhorsky et. al., *Detection of high Z objects using multiple scattering of cosmic-ray muons*, Rev. Sci. Instrum. vol. 74 no. 10, p. 4294-4297, 2003

[7] L. Schultz et. al., *Imaga reconstruction and material z discriminaton via cosmic-ray muon radiography*, Nucl. Instrum Meth. A vol. 519, p.687-694, 2004

[8] S. Eidelman et. al., *Review of particle physics*, Phys. Lett. vol. B592, p.1, 2004

[9] A. Dempster, N. Laird, D. Rubin, J. R. Stat. Soc. B39, p. 1-78, 1977

[10] Y. Vardi, L. Shepp, L. Kaufman, J. Am. Stat. Assoc. 80, p. 8-20, 1985

[11] J. Kremer et. al., Phys. Lett. 83, p4241, 1999

[12] Rossi B., *High Energy Particles*, Englewood Cliffs, NJ: Prentice-Hall, 1952

[13] Larry J. Schultz, Gary S. Blanpied, Konstantin N. Borozdin, Andrew M. Fraser, Nicolas W. Hengartner, Alexei V. Klimenko, Christopher L. Morris, Chris Orum, Michael J. Sossong, *Statistical Recontruction for Cosmic Ray Muon Tomography*, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 16, NO. 8 ,2007

[14] S. Presente, S. Vanini, M. Benettoni, P. Calvini, P. Checchia, E. Conti, F. Gonella, G. Nebbia, S. Squarcia, G. Viesti, A. Zenoni, G. Zumerle, *First result on material identification and imaging with a large-volume muon tomography prototype*, Nuclear Instrument and Methods in Physics Research, Elsevier, 2009

[15] Brian W. Kernighan, Dennis M. Ritchie *Il linguaggio C - Principi di programmazione e manuale di riferimento*, Pearson Education, 1988

[16] Philip Machanick, *C and C++ in 5 days*, South Africa. Computer Science Department University of the Witwatersrand, 1994

[17] `http://wwwcdf.pd.infn.it/valgrind/ms_main.html`

# Ringraziamenti

Durante questa tesi e gli ultimi due anni di corsi sono molte le persone che dovrei ringraziare, ma in particolare mi sembra giusto dedicare un ringraziamento particolare ad alcune.

Ringrazio il mio relatore Giancarlo Calvagno per avermi sempre dato fiducia sul mio operato in modo incondizionato e per essere sempre stato presente e disponibile.
I correlatori Simone Milani e Sara Vanini per tutto il grande supporto che mi hanno dato durante questo periodo.
Matteo Furlan del dipartimento di Fisica per essere sempre di grande consiglio e velocissimo a rispondere alle mie email con i più disparati dubbi.

I miei genitori per avermi sempre supportato in questi anni e per avermi dato questa grande possibilità.

DonTazza per le sue pause cicca, per i soliti noiosi discorsi infiniti da ingegneri, per le sue mille paranoie che rendevano i corsi più divertenti ma in particolare modo alle sue prestazioni vergognose.
Manu compagno di scuola per tre anni, di università per cinque, di stanza per due e di innumerevoli avventure padovane.
Panfi e Minky per essere stati dei buoni compagni di convitto.
Franco per tutti i mercoledì.
Ton per i viaggi da e verso Padova e per qualche mercoledì di meno.
Giulia perchè se devo litigare con qualcuno è sempre meglio farlo con lei.

Laura per avermi fatto compagnia un pomeriggio in pronto soccorso quando sicuramente avrebbe avuto di meglio da fare.
Giulietta anche se non so spiegare bene il perchè.

La mia migliore amica per avermi offerto sempre un grande supporto morale ogni volta che ne ho avuto bisogno, per avere sempre la pazienza di ascoltarmi e di darmi comunque saggi consigli anche se non li ascolto mai.

Inoltre ringrazio: Pino, Lord V, Possa, Buosi, Omic e Zoggia.