DOTTORATO DI RICERCA

in

SCIENZE COMPUTAZIONALI E INFORMATICHE

Ciclo XVIII

Consorzio tra Università di Catania, Università di Napoli Federico II,
Seconda Università di Napoli, Università di Palermo, Università di Salerno

SEDE AMMINISTRATIVA: UNIVERSITÀ DI NAPOLI FEDERICO II

Daniela Casaburi

Ricostruzione e segmentazione di immagini 3D: dal modello allo sviluppo del software in ambiente di calcolo parallelo.

TESI DI DOTTORATO DI RICERCA

Premessa

Nella società moderna le immagini rivestono un ruolo sempre più importante: l'*immagine* è uno strumento potente e ampiamente usato nella comunicazione ed anche un mezzo semplice, compatto e diffuso per la rappresentazione del mondo fisico. Anche nel mondo scientifico ed in particolare nella CSE (**C**omputational **S**cience and **E**ngineering) l'*immagine* riveste un ruolo fondamentale essendo alla base della simulazione e la visualizzazione di grandi quantità di dati.

Appare allora evidente il crescente interesse per le tecniche e i metodi numerici che consentono di rappresentare un'immagine e permettono di migliorarla o estrarne informazioni. Tali tecniche sono alla base dell'*Image Processing* e dell'*Image Analysis*.

Fino a pochi anni fa le tecniche utilizzate erano basate su metodi euristici ed approcci *"ad hoc"*. La svolta significativa nell'Image Processing e Analysis si è avuta con l'introduzione di modelli e metodi matematici.

Alla base di tali modelli c'è la classe dei *Problemi Inversi*, problemi in cui a partire dalle note caratteristiche (f) dello strumento ottico utilizzato nell'acquisizione (telecamera, macchina ecografica etc.), dal rumore random additivo k e dall'immagine degradata z, con

$$z = f(u) + k,$$

si vuole ricostruire un'approssimazione $\tilde{u}$ della soluzione ideale

$$u = f^{-1}(z - k)$$

in modo tale che

$$||\tilde{u} - z|| < \epsilon,$$

con ϵ opportuna.

Una caratteristica di tali problemi è la perdita di informazioni significative nel passare dal dato u al risultato z; ad esempio, si passa dalla scena reale u nello spazio 3D all'immagine acquisita z definita nello spazio 2D, con un'evidente perdita di informazioni.

Per compensare la perdita di informazioni occorre assegnare informazioni aggiuntive, occorre cioè regolarizzare il problema inverso. A partire dall'idea di Tikhonov nel 1977 sono stati introdotti molti operatori di regolarizzazione con l'intento di fornire modelli matematici sempre più affidabili e aderenti alle caratteristiche effettive di un'immagine.

Definito il modello matematico ($M(P)$) il suo effettivo utilizzo in applicazioni concrete necessita la messa a punto di metodi numerici per la realizzazione del corrispondente problema discreto $M_h(P)$, di algoritmi e software efficienti che permettano di ottenere la soluzione desiderata in tempo utile. È a questo livello che si inserisce il mio lavoro di tesi, nel quale, partendo dall'equazione di diffusione del flusso a curvatura media, assunta come modello matematico del problema di ricostruzione e segmentazione di una immagine, vengono analizzati, discussi e implementati tutti i passi computazionali necessari allo sviluppo dell'elemento di software in un ambiente di calcolo parallelo ad alte prestazioni.

Dopo una breve introduzione ai problemi del denoising e della segmentazione d'immagini 3D, nel *Capitolo 1* si illustra il legame presente tra le PDE e l'Image Processing ed Analysis, ponendo particolare attenzione ai modelli di diffusione non lineari (Modelli di

flusso a curvatura media - Modelli Level Set). Nel *Capitolo 2* sono analizzati nel dettaglio il denoising e la segmentazione di immagini 3D: dal modello matematico fino alla loro discretizzazione. Nel *Capitolo 3* è descritto il metodo numerico utilizzato per la risoluzione del nucleo computazionale di base: il **JFNK** (**J**acobian-**F**ree **N**ewton-**K**rylov). Nel *Capitolo 4* è introdotto l'ambiente di sviluppo dell'algoritmo: la libreria *PETSc* (**P**ortable, **E**xstensible **T**oolkit for **S**cientific **C**omputation). Nel *Capitolo 5* vengono descritti i dettagli implementativi del software parallelo e della sua applicazione ad immagini mediche ecografiche. In fine nel *Capitolo 6* vengono illustrati alcuni esperimenti.

Lo scienziato non è l'uomo che
fornisce le vere risposte; è quello
che pone le vere domande.

C. Levi-Strauss

"... Ai miei genitori... "

É il momento di fermarmi e far scorrere le esperienze vissute negli ultimi tre anni. Non nego che a tal pensiero è subito spuntato un sorriso; questo vuol dire che sono contenta di quello che ho fatto, di quanto sono cresciuta sia da un punto di vista culturale che umano. Ma tutto ciò è stato possible anche grazie all'aiuto di alcune persone a cui è doveroso da parte mia dire grazie.

Il primo Grazie al Prof. Almerico Murli che ha seguito i miei passi nella strada della formazione, non perdendo mai di vista il mio carattere e la persona che sono; ha infatti saputo mettermi sotto pressione nella giusta misura portandomi a dare il meglio; ha inoltre saputo leggere ed indirizzare le mie potenzialità nella direzione migliore.

Un Grazie particolare alla Prof. Luisa D'Amore che ha avuto fiducia nelle mie capacità e mi ha spronato a vivere esperienze importanti aiutandomi nei momenti "critici".

Un grazie va a chi nei momenti non semplici ha saputo dirmi .."Dai ancora un pò!".., o mi ha regalato un sorriso, o semplicemente ha capito e sorvolato sui miei modi bruschi dovuti alla non tranquillità della mia testa. Un grazie è doveroso a chi si è soffermato ad ascoltarmi e ha insinuato domande nella mia testa, a chi nei momenti di allarme è stato presente.

Un grazie alle mie sorelle e a mio cognato che non mi hanno mai

fatto sentire sola nelle mie scelte credendo in me e portandomi a ragionare quando occorreva.

Un grazie ai mie genitori che sono sempre stati al mio fianco e mi hanno trasmesso il senso del dovere e l'importanza di fare sempre al meglio delle proprie possibilità. É bello vedere negli occhi di mia madre la gioia e la soddisfazione per quello che sono riuscita a fare. Grazie a chi già tre anni fa era a dir poco entusiasta e orgoglioso di ciò che oggi sto per concludere.

Grazie di cuore

Daniela

Indice

Premesse

Indicando con u l'immagine reale e con z l'immagine acquisita, un modello di degradazione dell'immagine z è del tipo

$$M(P): \quad z = f(u) + k \tag{1}$$

dove k indica il rumore introdotto nell'acquisizione di z ed f la funzione che dipende dallo strumento di acquisizione considerato.

Definizione: *Ricostruzione d'immagine.*
Si definisce ricostruzione di un'immagine l'operatore inverso

$$M^{-1}(P) : z \to \tilde{u}$$

laddove $\tilde{u}$ indica un'approssimazione di u, tale che:

$$||\tilde{u} - z|| < \epsilon$$

con ϵ opportuno.

Nelle Figure 1 e 2 sono rappresentate rispettivamente un'immagine sintetica priva di rumore e con rumore additivo.

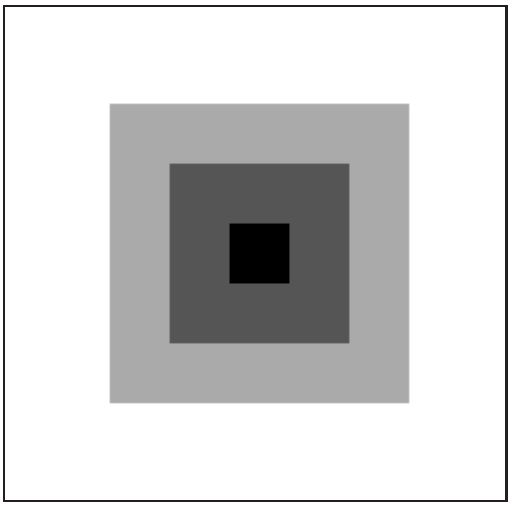

Figura 1: Immagine $z = f(u)$ con $z \equiv 0$.

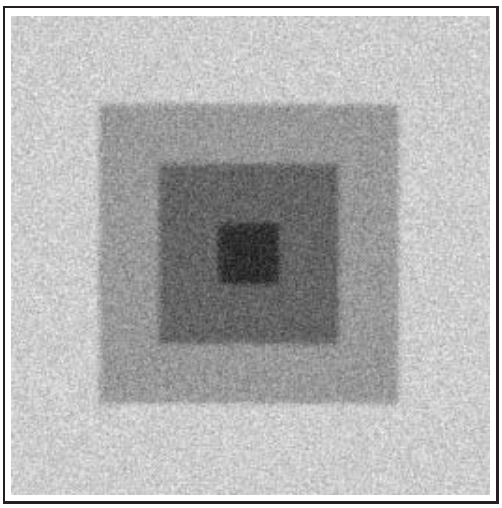

Figura 2: Immagine $z = f(u) + k$.

Dato il modello di degradazione (1) $M(p)$, diamo la seguente

Definizione [4]: *Segmentazione d'immagine (1).*
Si definisce segmentazione dell'immagine z la partizione di z nelle sue parti "costituenti".

La definizione sopra enunciata è ambigua e troppo generica, in quanto dipende dal tipo di informazioni che si vuole estrarre dall'immagine stessa (ovvero le parti *"costituenti"*). I due principali obiettivi per cui si fa la segmentazione sono: determinare le regioni *"omogenee"* che costituiscono l'immagine z (Figure 3 e 4); estrarre dall'immagine alcuni contorni *"significativi"* che assomiglino ad una curva assegnata (Figure 5 e 6).

In entrambi i casi i "contorni" ricoprono un ruolo fondamentale e generano le problematiche seguenti: *"come può essere rappresentato un contorno?"* e *"quale criterio ci garantisce di selezionare i contorni dell'immagine e non quelli prodotti da eventuale rumore?"*.

Al fine di precisare la definizione di segmentazione diamo le seguenti definizioni:

Definizione: *Immagine*
Un'immagine è rappresentata da una funzione reale a valori reali

$$u^0 : x \in \Omega \rightarrow u^0(x) \in \Re$$

dove $x = (x_1, \ldots, x_d) \in \Omega \subset \Re^d$ indica le coordinate del punto rappresentativo del pixel/voxel della griglia Ω di dimensione $l_1 \times l_2 \times \ldots \times l_d$ sulla quale è definita la funzione u^0.

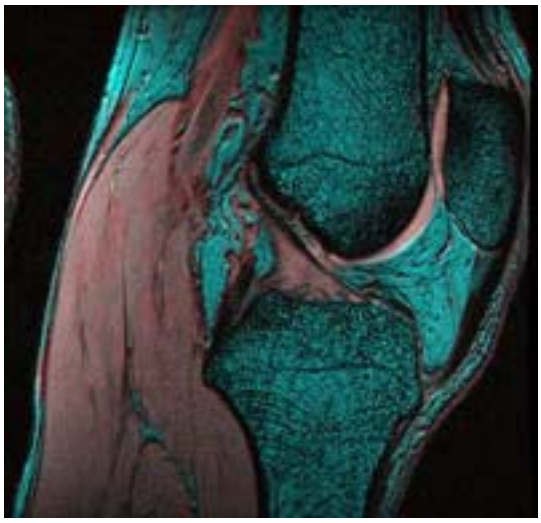

Figura 3: In figura è rappresentata l'immagine iniziale z.

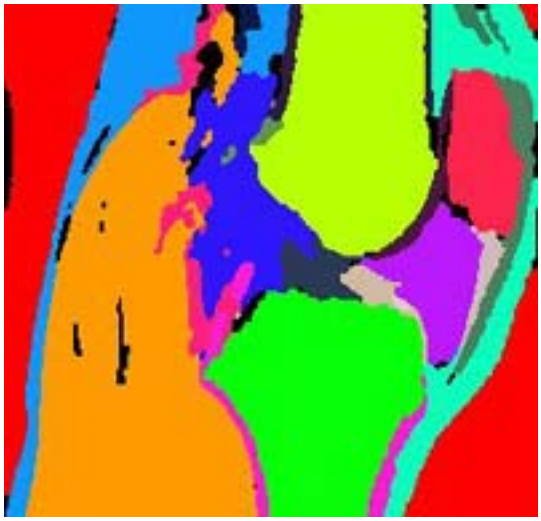

Figura 4: In figura è rappresentata l'immagine z segmentata.

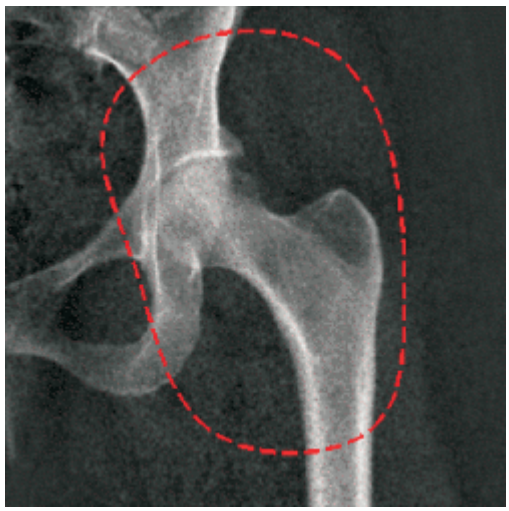

Figura 5: In figura è rappresentata l'immagine assegnata e la curva iniziale (linear tratteggiata in rosso).

Figura 6: In figura è rappresentata l'immagine assegnata e la curva ottenuta (linear tratteggiata in rosso) a partire da quella iniziale.

Definizione [4]: *Edge di un'immagine*
$\bar{x}$ è un punto di edge della funzione u se

$$\frac{\partial u}{\partial x}(\bar{x}) = \max_x \frac{\partial u}{\partial x}(x)$$

Se u è almeno C^3 in $\bar{x}$, si ha che:

$$\frac{\partial^2 u}{\partial x^2}(\bar{x}) = 0 \quad e \quad \frac{\partial^3 u}{\partial x^3}(\bar{x}) \leq 0.$$

Tale definizione nasce dal fatto che la precisione visiva di un contorno avviene in presenza di una variazione brusca dell'intensità dell'immagine $u(x)$ tra lo sfondo e l'oggetto. Poichè in questo caso l'intensità del gradiente di u è molto alta sul contorno dell'oggetto si sceglie $|\nabla u|$, o una funzione di esso, come *"identificatore"* di contorni. Quindi, assegnata un'immagine $u \in \Re^n$ definiamo omogenee le regioni in cui $|\nabla u(x)|$ è lo stesso, a meno di un errore dell'ordine di ϵ (con $\epsilon \to 0$). Le regioni omogenee sono delimitate da curve Γ in cui il valore del $|\nabla u|$ non è confrontabile con quello assunto internamente alla regione omogenea. La curva Γ si può, quindi, vedere come un *"contorno"* ovvero una curva di *"separazione"* tra due o più regioni omogenee (In Figura 7 è fornito un esempio di regioni omogenee nel caso in cui $u \in \Re^2$).

Possiamo ora riformulare la definizione di segmentazione di un'immagine.

Definizione: *Segmentazione d'immagine (2).*
Assegnata l'immagine u, definita sul dominio Ω:

$$u : \Omega \longrightarrow \Re$$

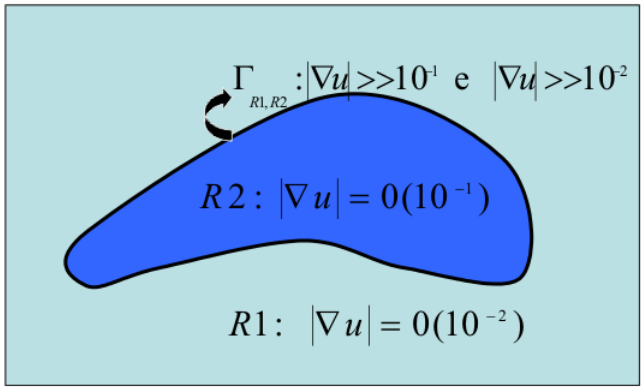

Figura 7: In figura sono rappresentate le due superfici omogenee $R1$ ed $R2$ e la loro superficie di separazione Γ. In $R1$ $|\nabla u| = O(10^{-2})$, in $R2$ $|\nabla u| = O(10^{-1})$ ed in Γ ha un ordine di grandezza non confrontabile con i due precedenti.

con

$$\Omega = \bigcup_i \Omega_i$$

tale che

$$\Omega_i \cap \Omega_j \quad se\ i \neq j,$$

si vuole determinare la curva Γ*:*

$$\Gamma = \bigcup_i (\partial\Omega_i) \cap \Omega.$$

curva di separazione di regioni "omogenee".

Il modello matematico considerato in questo lavoro, per la descrizione del problema di denoising e di segmentazione, si basa sull'equazione di flusso a curvatura media [20, 14, 15]. Tale modello, a partire dalla caraterizzazione di una curva come curva di livello

(introdotta da S. Osher e J. A. Sethian [39]),

> *"A curve can be seen as the zero-level*
> *of a function in higher dimension."*

è alla base della segmentazione e della ricostruzione di immagini attraverso l'analisi del movimento di una curva in funzione delle sue curvature.

L'equazione di flusso a curvatura media rientra nella forma:

$$\begin{cases} \frac{\partial u(x(t),t)}{\partial t} + F(x, u(x,t), \nabla u(x,t), \nabla^2 u(x,t)) = 0 & in \quad (0,T) \times \Omega \\ u(0,x) = u_0(x) & in \quad \Omega \end{cases} \tag{2}$$

dove $F(\cdot)$ rappresenta l'operatore differenziale del secondo ordine, $u(x,t)$ la funzione calcolata a partire dalla condizione iniziale $u_0(x)$, di classe almeno C^4 rispetto alla componente spaziale e C^2 rispetto a quella temporate, ∇u e $\nabla^2 u$ il gradiente e la matrice Hessiana di u rispetto alla variabile spaziale x.

Nell'espressione (2) un ruolo fondamentale è ricoperto dal parametro t. Partendo dall'immagine iniziale $u_0(x)$ e risolvendo la (2), si costruisce una famiglia di funzioni $\{u(x,t)\}_{t>0}$ in cui ogni elemento rappresenta un'approssimazione di $u_0(x)$. Il parametro t è denominato parametro di scala.

Nella (2) la scelta dell'operatore F è determinante e deve verificare due condizioni apparentemente contraddittorie: la prima è che $u(x,t)$ fornisca un'approssimazione di $u_0(x)$ a variazione limitata ovvero che ∇u sia sufficientemente *"smooth"*; la seconda è quella di preservare alcune caratteristiche di u, ad esempio gli edges, che sono

proprio i punti di significativa variazione della funzione e/o del ∇u.

Al fine di darne una seppur breve descrizione bisogna premettere alcuni concetti relativi all'analisi multiscala.

Capitolo 1

Le PDE e l'Image Processing ed Analysis

1.1 L'Analisi Multiscala delle immagini

Qualunque oggetto del mondo fisico è caratterizzato da una sua dimensione ed abbiamo bisogno di uno strumento (occhio, telecamera, etc. ..) per visualizzarlo in tutti i suoi *"dettagli"*. Definiamo *"scala"* l'intervallo di visibilità che lo strumento può visualizzare.
Una qualunque osservazione del mondo fisico è quindi condizionata dallo strumento ottico utilizzato e dalla scala che esso può rappresentare. Per diminuire o ampliare, ad esempio, la scala dell'occhio umano vengono utilizzati strumenti quali i microscopi ed i telescopi. Questo implica che non potremmo mai osservare la realtà fisica ma solo qualche cosa di molto prossima ad essa; che possiamo parlare e definire risoluzioni infinite del mondo esterno, ma sono pure astrazioni matematiche.

Definizione 1.1 *: Scala Interna (SI)*
Fissato uno strumento ottico, definiamo sua scala interna il più piccolo dettaglio visibile.

Definizione 1.2 *: Scala Esterna (SE)*
Fissato uno strumento ottico, definiamo sua scala esterna il più grande dettaglio che può essere visualizzato.

L'analisi multiscala di un oggetto è costituita dall'insieme di rappresentazioni di un oggetto reale in tutte le scale comprese tra quella interna e quella esterna (Figura 1.1).

1.2 Modelli anisotropici di diffusione non lineare

Il primo passo per l'uso delle PDE per l'analisi multiscala nell'Image Processing è stato fatto negli anni ottanta [31, 50]. A partire dall'osservazione che la funzione gaussiana

$$G_\sigma = \frac{1}{(4\pi\sigma)^{d\backslash 2}} \cdot e^{|x|^2/4\sigma}$$

è una soluzione fondamentale dell'equazione del calore

$$\begin{cases} \frac{\partial u(x(t),t)}{\partial t} = c \cdot \triangle u(x(t),t) \\ u(x,0) = u^0(x) \end{cases} \tag{1.1}$$

è stato possibile sostituire alla classica operazione sull'immagine u - convoluzione di u con G_σ con una assegnata varianza $v = \sqrt{2\sigma}$ - la risoluzione dell'equazione del calore per un corrispondente tempo $t = \sigma$ e condizione iniziale coincidente con l'immagine u^0.

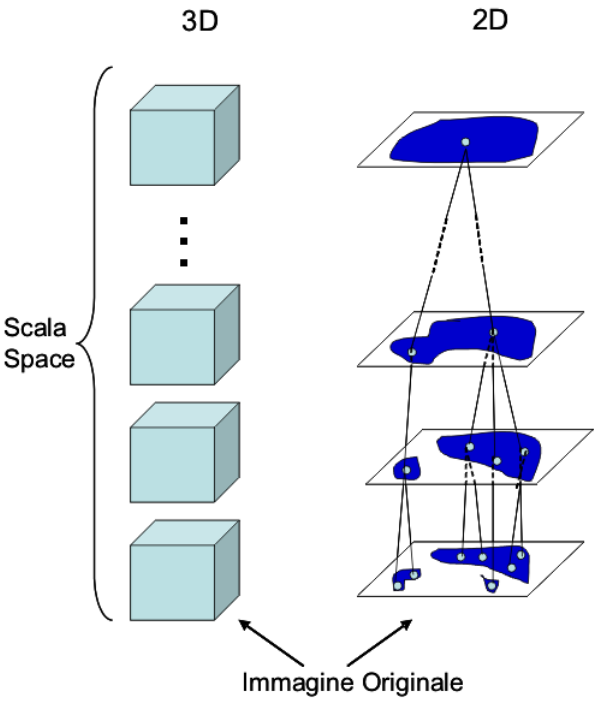

Figura 1.1: In figura è rappresentato un esempio di profondità della struttura dell'immagine nello scala-space, in termini di voxel e di pixel.

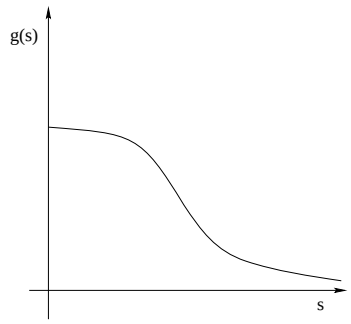

Figura 1.2: In figura è rappresentato un esempio di grafico della funzione $g(s)$

I pionieri nell'uso di operatori differenziali non lineari sono stati Perona e Malik [40], i quali hanno introdotto nell'operatore differenziale (1.1) un termine dipendente dal gradiente di u, in grado di gestire e seguire la riduzione del gradiente stesso. In particolare hanno proposto l'equazione non lineare:

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (g(|\nabla u|)\nabla u) = 0 \\ u(0) = u_0 \end{cases} \tag{1.2}$$

La funzione g (es. in Figura 1.2) deve essere regolare, non crescente e

$$\begin{cases} g(0) = 1 \\ g(s) \geq 0 \\ \lim\limits_{s \to \infty} g(s) = 0 \end{cases}$$

Il processo di regolarizzazione ottenuto con l'equazione (1.2) è *"con-*

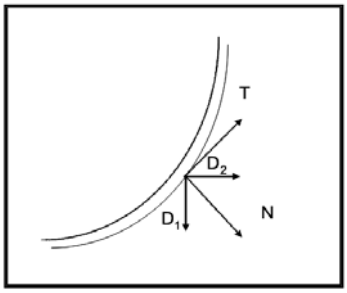

Figura 1.3: *In figura sono rappresentate le componenti del Laplaciano di $u(x)$ in un processo isotropico.*

dizionato", nel senso che:

$$\begin{cases} g(|\nabla u|) \longrightarrow 0 & se \quad |\nabla u| \longrightarrow \infty \\ g(|\nabla u|) \longrightarrow \infty & se \quad |\nabla u| \longrightarrow 0 \end{cases}$$

La (1.2) è nota come *equazione di diffusione anisotropica*[1], in quanto il coefficiente $c(x,t) \equiv g(s,t)$ dipende dal gradiente di u e quindi dalla variazione di intensità.
La (1.2) è stata poi sostituita dalla (1.3)

$$\frac{\partial u}{\partial t} - \nabla \cdot (g(|\nabla G_\sigma * u|)\nabla u) = 0 \tag{1.3}$$

per far fronte a problemi di mal posizione della (1.2) [12].

La (1.2) e la (1.3) sono primi esempi dell'analisi multiscala delle immagini [1, 3]. In particolare in [2] Alvarez, Lions e Morel hanno proposto una classe di equazioni differenziali paraboliche non lineari:

[1] Un processo di diffusione è definito isotropico se l'evoluzione del processo stesso risulta uniforme in ogni direzione, in caso contrario è definito anisotropico [4]. In Figura 1.3 è illustrato come evolvano le componenti del laplaciano di u, definito nello spazio 2D, in un processo isotropico. Da un punto di vista geometrico si può affermare che: fissato B, un insieme di elementi di $\Re^2$, si definisce isotropico se, $\forall b \in B$ si ha che $Rb \in B$, per ogni isometria R di $\Re^2$ [24].

$$\begin{cases} \frac{\partial u}{\partial t} = |\nabla u| \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) \\ u(x,0) = u_0(x) \end{cases} \tag{1.4}$$

in cui $u_0(x)$ rappresenta l'immagine iniziale.

1.3 I modelli di flusso a curvatura media e i modelli Level-Set

Il *metodo level-set* è una tecnica numerica utilizzata per la determinazione delle superfici di separazione in un'immagine. Nel seguito si dà una breve descrizione del metodo.

Assegnata una superficie di separazione Γ in $\Re^n$ che delimita una regione aperta ed omogenea Ω, si vuole studiare il suo spostamento e il relativo campo velocità $\vec{v}$ (Figura 1.4). Sia $x = x(x_1, x_2, \ldots, x_n) \in \Re^n$, la funzione *"level set"* φ deve soddisfare le seguenti proprietà (Figura 1.5):

$$\begin{aligned} \varphi(x,t) &> 0 \quad for \quad x \in \Omega \\ \varphi(x,t) &< 0 \quad for \quad x \notin \bar{\Omega} \\ \varphi(x,t) &= 0 \quad for \quad x \in \partial\Omega = \Gamma(t) \end{aligned}$$

La superficie Γ viene quindi definita ad ogni t, attraverso la localizzazione dell'insieme dei punti in cui φ si annulla. L'equazione che descrive il moto di φ, prodotto da $\vec{v}$, è la seguente:

$$\frac{\partial \varphi}{\partial t} + \vec{v} \cdot \nabla \varphi = 0. \tag{1.5}$$

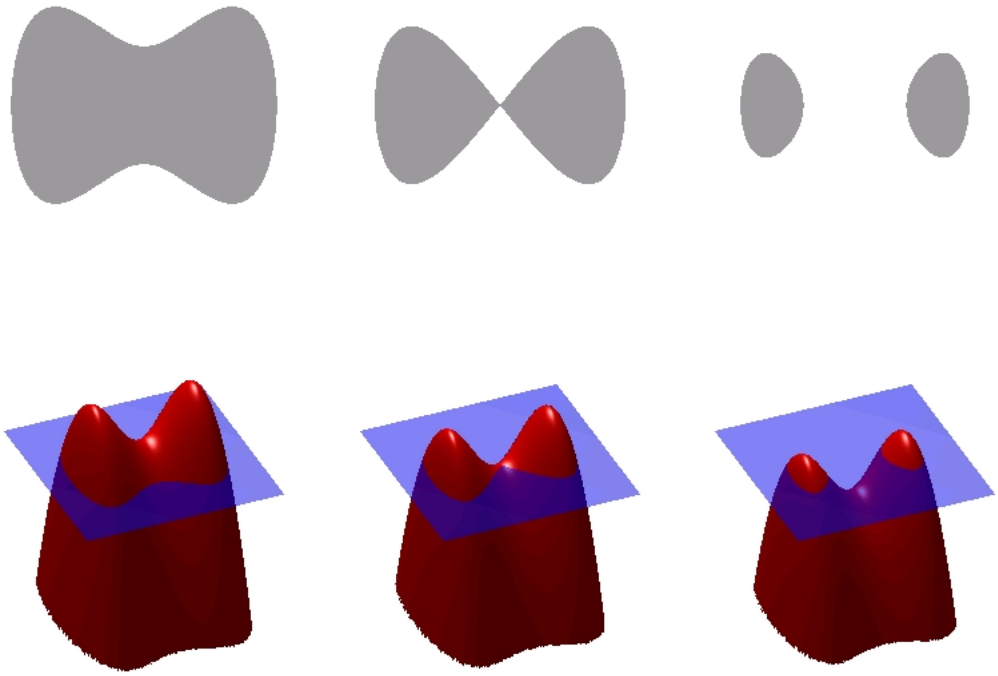

Figura 1.4: La figura illustra l'idea alla base del metodo level set. Partendo dall'angolo in alto a sinistra è illustrata l'evoluzione della sagoma di una regione limitata con un contorno ben delineato. Di seguito, la superficie in rosso rappresenta il grafico di una funzione level set φ che determina la sagoma sopra indicata; la regione blu rappresenta, invece, il piano $x - y$. Il contorno della sagoma in grigio è la curva di livello zero di φ, mentre la sagoma è essa stessa l'insieme dei punti del piano per i quali φ è positiva o nulla.

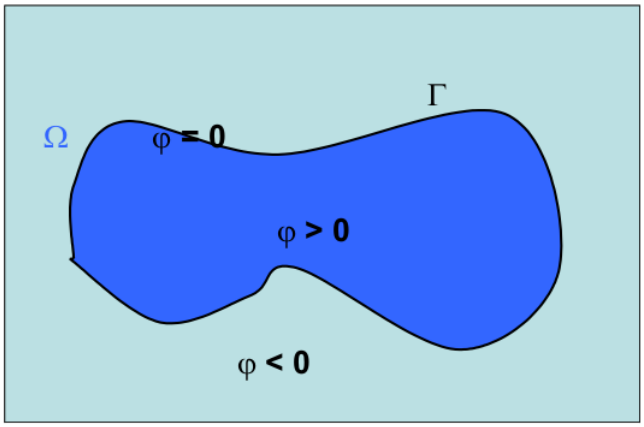

Figura 1.5: In figura è rappresentato il valore assunto dalla funzione $\varphi(x,t)$ internamente alla regione aperta Ω ($\varphi(x,t) > 0$), esternamente alla chiusura di Ω ($\varphi(x,t) < 0$) e sulla curva $\Gamma = \partial\Omega$ ($\varphi(x,t) = 0$).

Considerando solo la componente normale di $\vec{v} = v_N\vec{N} + v_t\vec{T}$ la (1.5) diventa:

$$\frac{\partial\varphi}{\partial t} + v_N\vec{N}\cdot\nabla\varphi = 0, \tag{1.6}$$

essendo verificate le uguaglianze

$$\vec{N}\cdot\nabla\varphi = \frac{\nabla\varphi}{|\nabla\varphi|}\cdot\nabla\varphi = \frac{|\nabla\varphi|^2}{|\nabla\varphi|} = |\nabla\varphi|$$

la (1.6) può essere riscritta come:

$$\frac{\partial\varphi}{\partial t} + v_N|\nabla\varphi| = 0 \tag{1.7}$$

Nel caso in cui v_N è funzione della direzione normale unitaria, la (1.6) diventa l'equazione del primo ordine di Hamilton-Jacobi:

$$\frac{\partial\varphi}{\partial t} + |\nabla\varphi|\gamma(\vec{N}) = 0$$

dove $\gamma = \gamma(\vec{N})$ è un'assegnata funzione della normale $\vec{N} = \frac{\nabla\varphi}{|\nabla\varphi|}$.

Definizione 1.3 *: Curvatura di funzione in $\Re^2$*
Assegnata la funzione reale u di calsse C^2, definita in $\Omega \subset \Re^2$, tale che $\nabla u \neq 0$. Si definisce curvatura di u la funzione che ad ogni $x \in \Omega$ associa il valore reale:

$$curv_u(x) = \frac{1}{|\nabla u|^3} \cdot \nabla u(\nabla u^{\perp}, \nabla u^{\perp})$$

avendo inidcato con ∇u il gradiente di u e con $\nabla u^{\perp}$ il suo vettore ortogonale.

Definizione 1.4 *: Curvature principali di una superficie.*
Assegnata la superficie M, si indichi con P un suo punto, con $T_P M$ il piano tangente ad M in P e con N il vettore per P perpendicolare a $T_P M$ e rivolto verso l'esterno. Si consideri il fascio di piani normali ad M in P e denominiamo sezioni normali di M in P ($\{Sez_M(P)\}_{Piano\_Normale}$) le curve piane lungo le quali i piani del fascio intersecano la superficie M.
Definifiamo curvature princiali della superficie M in P:

$$k_1 = \max_{Piano\_Normale} \{Sez_M(P)\}$$
$$k_2 = \min_{Piano\_Normale} \{Sez_M(P)\}$$

Proposizione 1
Sia $u = u(x,t)$ una funzione il cui gradiente $\nabla u = (u_{x_1}, u_{x_2}, \ldots, u_{x_N})$ è non nullo in $Q \subseteq \Re^N \times (0,\infty)$. Se e solo se u verifica l'equazione

alle derivate parziali parabolica

$$\frac{\partial u(x(t),t)}{\partial t} = \left(\delta_{ij} - \frac{u_{x_i}u_{x_j}}{|\nabla u|^2}\right) u_{x_i x_j}$$

ogni curva di livello di u evolve lungo la propria curvatura media.

Dimostrazione:
Consideriamo la famiglia delle curve zero

$$\Gamma_t \equiv \{x \in \Re\ N | u(x,t) = 0\}$$

con $t \geq 0$*, ed il campo vettoriale di curvatura media di* $\vec{\boldsymbol{n}}$ *(campo vettoriale unitario normale di* $\{\Gamma_t\}_{t\geq 0}$ *in* Q*) dato da*

$$-\frac{1}{N-1}\ \nabla \cdot (\vec{\mathbf{n}})\vec{\mathbf{n}}$$

Fissando $x \in \Gamma_t \cap Q$*, si verifica che evolve secondo la ODE seguente*

$$\begin{cases} \dot{x}(s) &= -[\nabla \cdot (\vec{\mathbf{n}})\vec{\mathbf{n}}](x(s),s) \quad (s > t) \\ x(t) &= x \end{cases} \tag{1.8}$$

Essendo $x(s) \in \Gamma_s\ \ (s \geq t)$*, si ha* $u(x(s),s) = 0\ \ (s > t)$ *e quindi*

$$0 = \frac{\partial}{\partial s} u(x(s),s) = -[(\nabla u \cdot \vec{\mathbf{n}})\nabla \cdot (\vec{\mathbf{n}})](x(s),s) + \frac{\partial}{\partial t} u(x(s),s)$$

Ponendo $s = t$ *si ottiene*

$$\frac{\partial}{\partial t} u(x(t),t) = (\nabla u \cdot \vec{\mathbf{n}})\nabla \cdot (\vec{\mathbf{n}}) \qquad in\ \ (x,t).$$

Scegliendo quindi

$$\vec{\mathbf{n}} \equiv \frac{\nabla u}{|\nabla u|} \tag{1.9}$$

segue che in (x,t)

$$\frac{\partial u(x(t),t)}{\partial t} = |\nabla u| \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) = \left(\delta_{ij} - \frac{u_{x_i} u_{x_j}}{|\nabla u|^2} \right) u_{x_i x_j} \qquad (1.10)$$

In maniera simile si può dimostrare che la (1.10) è verificata in tutta la regione Q*.*

Viceversa:
Sia u *una soluzione della (1.10) nella regione* Q *in cui* $\nabla u \neq 0$*, si fissi* $t > 0$ *e* $x \in \Gamma_t \cap Q$ *soluzione della (1.8) e della (1.9). Se* u *verifica la (1.10) allora si deduce che*

$$u(x(s), s) = 0 \quad (s > t).$$

Di conseguenza la famiglia $\{\Gamma_t\}_{t \geq 0}$ *con*

$$\Gamma_t \equiv \{x \in \Re\, N | u(x,t) = 0\}$$

e tutte le curve di livello di u *evolvono in* Q *seguendo le proprie curvature medie.*

$\triangle$

Un esempio di modello del tipo (1.10) è il seguente:

$$\frac{\partial u(x(t),t)}{\partial t} = |\nabla u| \nabla \cdot \left(g^0 \, \frac{\nabla u}{|\nabla u|} \right) \qquad (1.11)$$

con g^0 funzione di Perona-Malik. Nell'equazione (1.11), proposta in [11, 10, 29]

$$|\nabla u| \approx |\nabla u|_\epsilon = \sqrt{\epsilon^2 + |\nabla u|^2}$$

si ottiene

$$\frac{\partial u}{\partial t} = \sqrt{\epsilon^2 + |\nabla u|^2}\nabla \cdot \left(g_\epsilon^0 \frac{\nabla u}{\sqrt{\epsilon^2 + |\nabla u|^2}} \right) \tag{1.12}$$

con

$$g_\epsilon^0 = g(|\nabla G_\sigma * I^0|)$$

dove $u(x,t)$ è la funzione incognita definita in $Q_T \equiv \Omega \times [0,T]$, $\Omega \subset \Re^d$ è un dominio limitato, $\partial\Omega$ rappresenta la frontiera Lipschitz continua, $[0,T]$ è l'intervallo di tempo, I^0 è un'immagine assegnata ed $\epsilon > 0$ è un parametro. All'equazione (1.12) si aggiungono le condizioni di Neumann al contorno:

$$\frac{\partial u}{\partial \nu} = 0 \quad in \;\; \partial\Omega \times [0,T]\,, \tag{1.13}$$

dove ν rappresenta la direzione normale a $\partial\Omega$, e la condizione iniziale:

$$u(x,0) = u^0(x) \;\; in \;\; \Omega \tag{1.14}$$

o le condizioni al contorno di Dirichlet:

$$u(x,t) = u^D \quad in \;\; \partial\Omega \times [0,T] \tag{1.15}$$

Senza ledere alla generalità si è posto $u^D = 0$.

Nella (1.12) la funzione $g : \Re_0^+ \longrightarrow \Re^+$ deve godere delle proprietà seguenti:

$$\begin{cases} non \;\; decrescente \\ g(0) = 1 \\ \lim\limits_{s\to\infty} g(s) = 0 \end{cases}$$

Ad esempio:

$$g(s) = \frac{1}{1 + Ks^2} \ \ con\ K \geq 0. \tag{1.16}$$

La funzione $G_\sigma(x) \in C^\infty(\Re^d)$ è la funzione Gaussiana con valor medio zero e varianza $\sqrt{\sigma}$:

$$G_\sigma(x) = \frac{1}{\sqrt{(4\pi\sigma)^d}} e^{-|x|^2/4\sigma}$$

mentre $\nabla G_\sigma * I^0$ rappresenta il prodotto di convoluzione:

$$\nabla G_\sigma * I^0 = \int_{\Re^d} \nabla G_\sigma(x - \xi)\hat{I}^0(\xi)d\xi$$

dove $\hat{I}^0$ è l'estensione di I^0 a tutto $\Re^d$ ottenuta per riflessione periodica. Il dominio $\Omega \subset \Re^d$ rappresenta, solitamente, un sotto dominio dell'immagine che contiene l'oggetto segmentato.

In questo lavoro si considera come modello matematico il problema descritto dalla (1.12). Il problema consiste nel determinare la funzione $u(x,t)$ nell'intervallo $[0,T]$ e sul dominio Ω, a partire dall'immagine I^0 e dalla condizione iniziale u^0.
Nel processo di denoising la condizione iniziale (1.14) coincide con l'intensità dei livelli di grigio dell'immagine stessa, ovvero $u^0 = I^0$. Da ciò scaturisce che il fattore $g(|\nabla G_\sigma * I^0|)$ non è costante e viene aggiornato adattativamente mediante l'espressione $g(|\nabla G_\sigma * u|)$.
Nella segmentazione di immagini la condizione iniziale u^0 rappresenta uno stadio iniziale della *"funzione di segmentazione"* $u(x,t)$. I^0 determina la funzione peso g per l'evoluzione della curvatura media della funzione di segmentazione $u(x,t)$, espressa mediante il metodo level-set.

Capitolo 2

Segmentazione e denoising di immagini 3D

2.1 Descrizione del metodo numerico

Per la discretizzazione delle derivate parziali nella (1.12) si sono considerati gli schemi alle differenze finite e quelli a volumi finiti, naturale evoluzione dei primi.

Si consideri la derivata di una funzione

$$u(x) : \Omega \in \Re \to \Re$$

in un punto

$$x_i = (i-1)h \;\; con \;\; h = \frac{L}{N-1} \;\; ed \;\; i = 1, \ldots, N$$

della griglia costruita sul dominio Ω

$$\frac{du}{dx}(x_i) = \lim_{h \to 0^+} \frac{u(x_i + h) - u(x_i)}{h} \tag{2.1}$$

Si consideri lo sviluppo in serie di Taylor della funzione u di punto

iniziale x_i e valutato in x_{i+1}

$$u(x_{i+1}) = u(x_i) + h\frac{du}{dx}(x_i) + \frac{1}{2}h^2\frac{d^2u}{dx^2}(x_i) + \frac{1}{6}h^3\frac{d^3u}{dx^3}(x_i) + O(h^4) \tag{2.2}$$

da cui:

$$\frac{du}{dx}(x_i) = \frac{u(x_{i+1}) - u(x_i)}{h} + O(h), \qquad 1 \leq i \leq N-1 \tag{2.3}$$

La (2.3) fornisce un'approssimazione corretta al primo ordine in h, ed è denominata *differenza finita in avanti* ("FD", **F**orward **D**ifference) (Figura 2.1, linea continua).
In maniera del tutto analoga, considerando lo sviluppo in serie di Taylor valutato in x_{i-1},

$$u(x_{i-1}) = u(x_i) - h\frac{du}{dx}(x_i) + \frac{1}{2}h^2\frac{d^2u}{dx^2}(x_i) - \frac{1}{6}h^3\frac{d^3u}{dx^3}(x_i) + O(h^4) \tag{2.4}$$

si ottiene la derivata prima alle *differenze divise all'indietro* ("BD", **B**ackward **D**ifference) (Figura 2.1, linea tratto e punto) al primo ordine

$$\frac{du}{dx}(x_i) = \frac{u(x_i) - u(x_{i-1})}{h} + O(h), \qquad 2 \leq i \leq N \tag{2.5}$$

Combinando insieme i due sviluppi di Taylor (2.2) e (2.4), in particolare sottraendo il secondo dal primo, si arriva invece all'espressione centrata, corretta al secondo ordine, denominata *differenza finita centrale* ("CD", **C**entral **D**ifference) (Figura 2.1, linea tratteggiata):

$$\frac{du}{dx}(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{2h} + O(h^2), \qquad 2 \leq i \leq N-1 \tag{2.6}$$

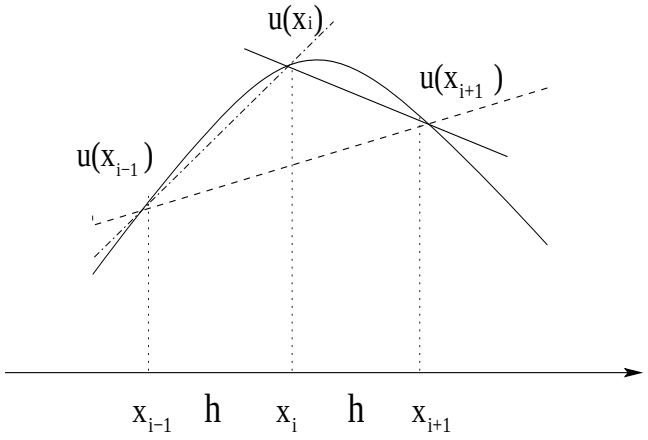

Figura 2.1: *In figura sono rappresentate le rette la cui pendenza è l'approssimazione alle differenze finite della derivata prima della funzione $u(x)$ valutata nel nodo x_i: all'indietro (linea tratto e punto), in avanti (linea continua) e centrata (linea tratteggiata).*

Si può notare come per le espressioni del primo ordine sia sufficiente che u sia di classe C^2, mentre l'espressione al secondo ordine richieda una regolarità maggiore, almeno C^4.

Considerando gli sviluppi in serie di Taylor (2.2) e (2.4) è possibile determinare anche un'approssimazione per la derivata seconda di u in x_i. In particolare si ha l'espressione seguente:

$$\frac{d^2u}{dx^2}(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h^2), \qquad 2 \leq i \leq N-1 \tag{2.7}$$

Nella (2.7) l'errore di troncamento è del secondo ordine in h^2 a patto che la funzione u sia dotata di derivata quarta.
L'insieme dei nodi $\{x_k\}_{k=1,\dots,N}$ che intervengono nella costruzione della derivata di u prende il nome di *"stencil"*. Gli schemi fin ora analizzati definiscono, quindi, stencil a 2 e 3 punti (approssimazioni

della derivata prima e della derivata seconda rispettivamente). Definendo la funzione u in $\Re^2$ si possono costruire esempi di stencil a 5 punti, come ad esempio quello relativo al calcolo del Laplaciano della funzione $u(x, y)$:

$$\nabla \cdot (\nabla u) = \nabla \cdot \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right) = \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] \tag{2.8}$$

Si consideri lo stencil a 5 punti relativa a $P(i, j)$ (Figura 2.2)

$$S_5 = \{w(i-1, j), e(i+1, j), n(i, j+1), s(i, j-1), P(i, j)\},$$

calcolando le derivate seconde con lo schema (2.7) si ha:

$$\begin{gathered} \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right]_P = \\ \left[\frac{u(e)-2u(P)+u(w)}{h^2} + \frac{u(n)-2u(P)+u(s)}{h^2} \right] + O(h^2) \end{gathered} \tag{2.9}$$

Indicando $u(p(i, j))$ con u_{ij}, la (2.9) si può scrivere come:

$$\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] = \left[\frac{u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}}{h^2} \right] + O(h^2)$$

Definendo la funzione u in $\Re^3$, la (2.8) risulta

$$\nabla \cdot (\nabla u) = \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right]$$

e per la sua discretizzazione si considera uno stencil a 7 punti (Figura 2.3)

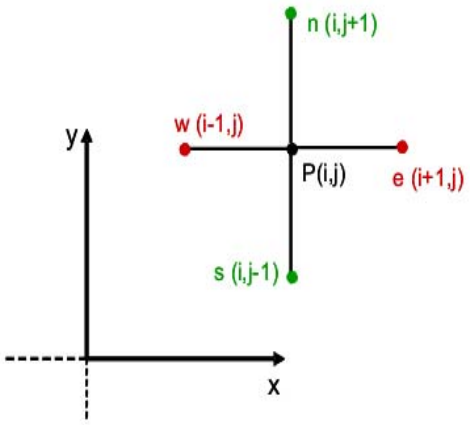

Figura 2.2: *In figura sono rappresentati i punti che costituiscono lo stencil S_5 relativa al punto $P(i,j)$*

$$S_7 = \{w(i-1,j,k), e(i+1,j,k), n(i,j+1,k), s(i,j-1,k), \\ b(i,j,k-1), t(i,j,k+1), P(i,j,k)\},$$

Si ha, quindi:

$$\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right] =$$

$$\left[\frac{u_{i+1,j,k}+u_{i,j+1,k}+u_{i,j,k+1}-6u_{i,j,k}+u_{i-1,j,k}+u_{i,j-1,k}+u_{i,j,k-1}}{h^2}\right] + O(h^2)$$

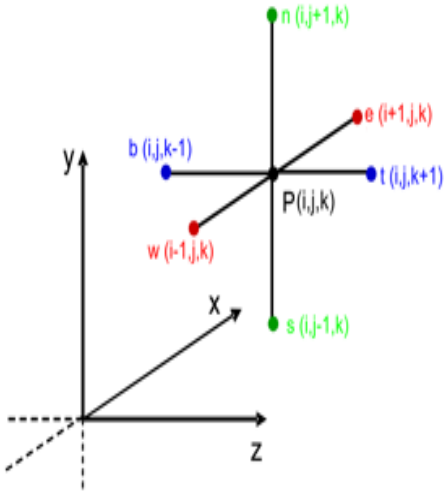

Figura 2.3: *In figura sono rappresentati i 6 punti adiacenti al punto P della griglia di voxel che individuano la stencil a 7 punti.*

2.1.1 Discretizzazione temporale semi-implicita.

Fissiamo due valori di t: t_1 e t_2, con $t_2 > t_1$. Per la discretizzazione delle derivate temporali $\frac{\partial u(x,t)}{\partial t}$ consideriamo lo schema alle differenze finite all'indietro (2.5); in particolare si ha:

$$\frac{\partial u(x,t)}{\partial t} = \frac{u(x,t_2) - u(x,t_1)}{t_2 - t_1} + O(h) = \frac{u^{(2)} - u^{(1)}}{h} + O(h)$$

avendo posto $h = t_2 - t_1$ e $u^{(j)} = u(x,t_j)$.
I termini non lineari ($|\nabla u(x,t)|$ e $g(|\nabla G_\sigma * u|)$) dell'equazione (1.12) sono calcolati in t_1 avendo considerato come passo corrente t_2,

$$|\nabla u(x,t)| = |\nabla u(x,t_1)| \tag{2.10}$$

e

$$g(|\nabla G_\sigma * u(x,t)|) = g(|\nabla G_\sigma * u(x,t_1)|) \tag{2.11}$$

mentre quelli lineari ($\nabla u(x,t)$) sono calcolati in t_2, ovvero

$$\nabla u(x,t) = \nabla u(x,t_2)$$

In generale, considerato l'intervallo $[0,T]$ e posto $t_0 = 0$, $t_N = T$ e $t_i = n\tau$, con $n = 1, \ldots, N$ e $\tau = \frac{T}{N}$, si ha:

Schema semi-implicito:
Assegnate I^0, l'immagine iniziale, ed u^0, la condizione iniziale, vogliamo determinare $u^n = u(x,t_n)$ soluzione dell'equazione:

$$\frac{1}{g\left(|\nabla G_\sigma * I^0|\right)|\nabla u^{n-1}|}\frac{u^n - u^{n-1}}{\tau} = \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|}\right) \qquad (2.12)$$

Posto:

$$g_\sigma^0 = g\left(|\nabla G_\sigma * I^0|\right)$$

otteniamo in forma compatta

$$u^n = F(u^{n-1})$$

con

$$F(u^{n-1}) = \frac{u^{n-1}}{I - \tau A(u^{n-1})}$$

e

$$A(u^{n-1}) = g_\sigma^0\, \nabla \cdot \frac{\nabla \bullet}{|\nabla u^{n-1}|}$$

Quindi, nello schema semi-implicito i termini non lineari sono calcolati sfruttando le informazioni ottenute alla scala precedente, mentre quelli lineari sono calcolati utilizzando le informazioni al passo corrente.

Lo schema semi-implicito descrive un processo di segmentazione se il termine

$$g_\sigma^0 = g\left(|\nabla G_\sigma * I^0|\right) \tag{2.13}$$

è costante e u^0 è denominata funzione iniziale di segmentazione; descrive un processo di denoising se $I^0 = u^0$, e quindi la funzione (2.13) al passo n assume la forma:

$$g_\sigma^{n-1} = g\left(|\nabla G_\sigma * u^{n-1}|\right)$$

2.1.2 Discretizzazione temporale implicita.

Nello schema di discretizzazione implicito sia i termini non lineari ($|\nabla u(x,t)|$ e $g(|\nabla G_\sigma * u|)$) dell'equazione (1.12) sia quelli lineari ($\nabla u(x,t)$) sono calcolati al passo corrente.

Schema implicito:
Assegnata I^0, l'immagine iniziale, ed u^0, la condizione iniziale, vogliamo determinare $u^n = u(x,t_n)$ soluzione dell'equazione:

$$\frac{1}{g\left(|\nabla G_\sigma * I^0|\right)|\nabla u^n|} \cdot \frac{u^n - u^{n-1}}{\tau} = \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^n|}\right) \tag{2.14}$$

Posto:

$$g_\sigma^0 = g\left(|\nabla G_\sigma * I^0|\right)$$

otteniamo in forma compatta

$$F(u^n, u^{n-1}) = 0$$

con

$$F(u^n, u^{n-1}) = \frac{1}{g_\sigma^0 \, |\nabla u^n|} \frac{u^n - u^{n-1}}{\tau} - \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^n|} \right)$$

Lo schema implicito descrive un processo di segmentazione se il termine

$$g_\sigma^0 = g\left(|\nabla G_\sigma * I^0|\right) \tag{2.15}$$

è costante e u^0 è denominata funzione iniziale di segmentazione; descrive, invece, un processo di denoising se $I^0 = u^0$, e quindi la funzione (2.15) al passo temporale n assume la forma:

$$g_\sigma^n = g\left(|\nabla G_\sigma * u^n|\right)$$

2.1.3 Discretizzazione spaziale a volumi finiti.

Nello schema ai volumi finiti la griglia coincide con la struttura di pixel/voxel dell'immagine, mentre i valori iniziali assegnati alla funzione u_0, e successivamente alla u_n, rappresentano un'approssimazione della media della funzione stessa lungo i pixel/voxel.

Diamo la definizione di alcune quantità che verranno utilizzate per la discretizzazione spaziale dell'equazione (2.14).
Sia τ_h la griglia di dimensione $l_1 \times l_2 \times l_3$ costruita sul dominio Ω; le celle (voxel) che costituiscono τ_h sono dette *volumi di controllo* (Figura 2.4); comunque presi due voxel p e q, con $p \neq q$, denotiamo

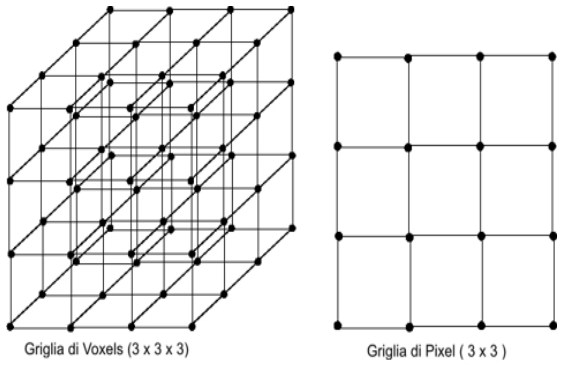

Figura 2.4: *Nella figura sono rappresentati un esempio di griglia di Voxel, coincidenti con i volumi di controllo, di dimensione* $3 \times 3 \times 3$ *e un esempio di griglia di Pixel, coincidenti con i pixel di controllo, di dimensione* 3×3

con e_{pq} la loro comune interfaccia. Denotiamo, poi, con $m(e_{pq})$ la misura di e_{pq} e con n_{pq} il vettore unitario e normale a e_{pq} orientato da p a q (Figura 2.5). Definiamo ε l'insieme delle coppie di volumi adiacenti, distinti e tali che la misura dell'interfaccia comune sia non nulla:

$$\varepsilon = \{(p,q) \in \tau_h \times \tau_h \mid p \neq q \;\; e \;\; m(e_{pq}) \neq 0\};$$

indichiamo, inoltre, con $N(p)$ l'insieme dei volumi di controllo in relazione con p appartenenti a ε (Figura 2.6)

$$N(p) = \{q \in \tau_h \mid (p,q) \in \varepsilon\}$$

e assumiamo che esistano famiglie di punti $(x_p)_{p \in \tau_h}$, con $x_p \in p$, per ogni $p \in \tau_h$, e che per ogni coppia $(p,q) \in \varepsilon$ sia verificata l'uguaglianza seguente:

$$\frac{x_q - x_p}{|x_q - x_p|} = n_{pq}.$$

Fissato il volume di controllo p indichiamo:

- la misura secondo Lebesgue di p con $m(p)$;

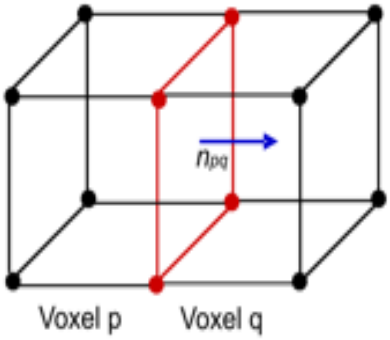

Figura 2.5: *Nella figura sono rappresentati 2 Voxel: p e q; in rosso è evidenziata la frontiera di e_{pq}, in blu il vettore unitario e normale n_{pq}.*

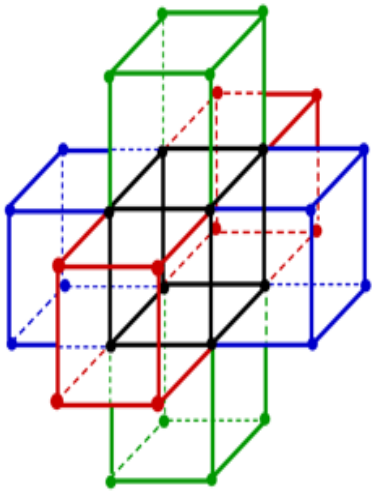

Figura 2.6: *Nella figura è rappresentato l'insieme $N(p)$, con p volume di controllo con tratto nero.*

- la frontiera di p con ∂p.
- il diametro con $\delta(p) = \underset{x_1,x_2 \in S}{max} |x_2 - x_1|$, con S insieme degli spigoli di p;

Al variare dei volumi p definiamo:

- con h il massimo dei diametri

$$h = \underset{p \in \tau_h}{max}\ \delta(p);$$

- con $d_{pq} = |x_q - x_p|$ la distanza Euclidea tra x_p e x_q;
- con x_{pq} il punto di intersezione di e_{pq} con il segmento $\overline{x_p x_q}$ (Figura 2.7);
- con T_{pq} indichiamo :

$$T_{pq} = \frac{m(e_{pq})}{d_{pq}}$$

- con $\bar{u}_p^n$ il valore rappresentativo di u^n nel volume finito p.

Effettuiamo la discretizzazione della (2.14), secondo lo schema ai volumi finiti. Da:

$$\int_p \frac{u^n - u^{n-1}}{|\nabla u^n| \cdot \tau}\, dx = \int_p \nabla \cdot \left(g\left(|\nabla G_\sigma * u^n|\right) \frac{\nabla u^n}{|\nabla u^n|} \right)\, dx \qquad (2.16)$$

discretizzando il termine a sinistra della (2.16):

$$\int_p \frac{u^n - u^{n-1}}{|\nabla u^n| \cdot \tau}\, dx = \frac{m(p)}{\tau \cdot |\nabla u_p^n|} \left(\bar{u}_p^{(n)} - \bar{u}_p^{(n-1)} \right) \qquad (2.17)$$

dove $|\nabla u_p^n|$ denota un'approssimazione del gradiente nel volume p e usando il *Teorema della Divergenza* sul termine a destra della (2.16)

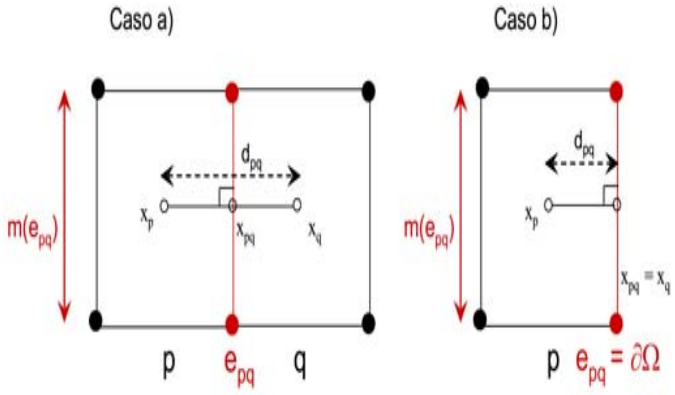

Figura 2.7: *Nella figura, fissato il volume di controllo p, sono rappresentati* d_{pq}, x_{pq} *ed* $m(e_{pq})$ *in presenza del pixel q adiacente (Caso A) e della frontiera di* Ω.

otteniamo:

$$\begin{aligned} &\int_p \nabla \cdot \left(g\left(|\nabla G_\sigma * u^n|\right) \frac{\nabla u^n}{|\nabla u^n|} \right) \, dx = \int_{\partial p} \frac{1}{|\nabla u^n|} \frac{\partial u^n}{\partial \nu} \, ds \\ &= \sum_{q \in N(p)} \int_{e_{pq}} g\left(|\nabla G_\sigma * u^n|\right) \frac{1}{|\nabla u_q^n|} \frac{\partial u^n}{\partial \nu} \, ds \end{aligned} \tag{2.18}$$

Discretizzando la derivata normale lungo la frontiera di p con il rapporto incrementale

$$\frac{\partial u^n}{\partial \nu} = \frac{\bar{u}_q^{(n)} - \bar{u}_p^{(n)}}{d_{pq}} + O(h) \;\; lungo \;\; e_{pq}$$

la (2.18) diventa:

$$\int_p \nabla \cdot \left(g\left(|\nabla G_\sigma * u^n|\right) \frac{\nabla u^n}{|\nabla u^n|} \right) dx = \sum_{q \in N(p)} \frac{1}{|\nabla u_p^n|} g_{pq}^{\sigma,n} T_{pq} (\bar{u}_q^{(n)} - \bar{u}_p^{(n)}) \tag{2.19}$$

con

$$g_{pq}^{\sigma,n} = g\left(|\nabla G_\sigma * \tilde{u}_{h,\tau}(x_{pq}, t_n)|\right)$$

essendo

$$\bar{u}_{h,\tau}(x,t) = \sum_{n=0}^{N} \sum_{p\in\tau} \bar{u}_p^n \chi_{\{x\in p\}} \chi_{\{t_{n-1}<t<t_n\}}$$

dove χ_A rappresenta la funzione caratteristica di A:

$$\chi_A = \begin{cases} 1 & se\ A\ \ vera; \\ 0 & altrimenti. \end{cases}$$

Quindi, sostituendo nella (2.16) la (2.17) e la (2.19) otteniamo:

$$\frac{m(p)}{\tau \cdot |\nabla u_p^n|}\left(\bar{u}_p^{(n)} - \bar{u}_p^{(n-1)}\right) = \sum_{q\in N(p)} \frac{1}{|\nabla u_q^n|}\, g_{pq}^{\sigma,n} T_{pq}(\bar{u}_q^{(n)} - \bar{u}_p^{(n)}) \quad (2.20)$$

e posto

$$b_p^n = \tfrac{m(p)}{|\nabla u_p^n|}$$

$$a_{pq}^n = \tfrac{1}{|\nabla u_q^n|}\, g_{pq}^{\sigma,n} T_{pq}$$

la (2.20) possiamo scriverla come:

$$b_p^n \bar{u}_p^{(n)} + \tau \sum_{q\in N(p)} a_{pq}^n \left(\bar{u}_p^{(n)} - \bar{u}_q^{(n)}\right) = b_p^n \bar{u}_p^{(n-1)} \quad (2.21)$$

che nella forma compatta $Ax = b$ risulta

$$\left(b_p^n + \tau \sum_{q\in N(p)} a_{pq}^n\right) \bar{u}_p^{(n)} = \tau \sum_{q\in N(p)} a_{pq}^n \bar{u}_q^{(n)} + b_p^n \bar{u}_p^{(n-1)}$$

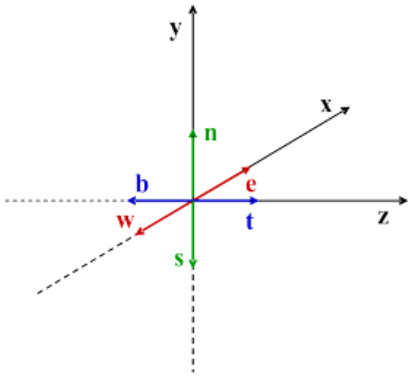

Figura 2.8: *Nella figura è rappresentato il sistema di assi cartesiani ortonormali XYZ con i relativi versori $\vec{e}$, $\vec{w}$, $\vec{n}$, $\vec{s}$, $\vec{t}$, $\vec{b}$.*

2.1.4 Discretizzazione spaziale alle differenze finite.

Come è solito fare per una griglia 3D di rettangoli, fissiamo un sistema di assi cartesiani ortonormali XYZ al quale associamo i versori opposti $\vec{e}$ e $\vec{w}$ per l'asse X, $\vec{n}$ e $\vec{s}$ per l'asse Y, $\vec{t}$ e $\vec{b}$ per l'asse Z (Figura 2.8).
Ad ogni spigolo P del voxel p assegnamo, quindi, una terma (i, j, k), dove i, j e k indicano le componenti del punto P rispettivamente lungo l'asse X, l'asse Y e l'asse Z del riferimento cartesiano fissato. Per ogni spigolo, inoltre, definiamo i 6 punti adiacenti nella griglia: con $w(i-1, j, k)$ indichiamo il **w**est di P, con $e(i+1, j, k)$ il suo **e**ast, con $s(i, j-1, k)$ il **s**outh, con $n(i, j+1, k)$ il **n**orth, con $b(i, j, k-1)$ il **b**attom e con $t(i, j, k+1)$ il **t**op (Figura 2.3). Tali punti sono utilizzati nello schema a sette punti delle differenze finite.

La funzione u^n valutata in un punto $P(i,j,k)$ della griglia la denoteremo $u^n_{i,j,k}$ e nei punti dell'insieme $C_P = \{w, e, s, n, b, t\}$ rispettivamente $u^n_{i-1,j,k}$, $u^n_{i+1,j,k}$, $u^n_{i,j-1,k}$, $u^n_{i,j+1,k}i$, $u^n_{i,j,k-1}$, $u^n_{i,j,k+1}$, omettendo, eventualmente, l'indice superiore n. Consideriamo la (2.21) nella forma:

$$\left(b^n_p + \tau \sum_{q \in N(p)} a^n_{pq} \right) \bar{u}^{(n)}_p - \tau \sum_{q \in N(p)} a^n_{pq} \bar{u}^{(n)}_q = b^n_p \bar{u}^{(n-1)}_p$$

Definiamo, quindi:

$$r_{i,j,k} = h^2 \bar{u}^{(n-1)}_{i,j,k}$$

e le quantità:

$$\begin{aligned}
z1^w_{i.j.k} &= \left(\tfrac{u_{i,j,k}-u_{i-1,j,k}}{h}\right)^2 + \left(\tfrac{u_{i,j+1,k}-u_{i,j-1,k}+u_{i-1,j+1,k}-u_{i-1,j-1,k}}{h}\right)^2 \\
&+ \left(\tfrac{u_{i,j,k+1}-u_{i,j,k-1}+u_{i-1,j,k+1}-u_{i-1,j,k-1}}{h}\right)^2 \\
z2^e_{i,j,k} &= \left(\tfrac{u_{i+1,j,k}-u_{i,j,k}}{h}\right)^2 + \left(\tfrac{u_{i+1,j+1,k}-u_{i+1,j-1,k}+u_{i,j+1,k}-u_{i,j-1,k}}{h}\right)^2 \\
&+ \left(\tfrac{u_{i,j,k+1}-u_{i,j,k-1}+u_{i+1,j,k+1}-u_{i+1,j,k-1}}{h}\right)^2 \\
z3^s_{i.j.k} &= \left(\tfrac{u_{i,j,k}-u_{i,j-1,k}}{h}\right)^2 + \left(\tfrac{u_{i+1,j,k}-u_{i-1,j,k}+u_{i+1,j-1,k}-u_{i-1,j-1,k}}{h}\right)^2 \\
&+ \left(\tfrac{u_{i,j,k+1}-u_{i,j,k-1}+u_{i,j-1,k+1}-u_{i,j-1,k-1}}{h}\right)^2 \\
z4^n_{i,j,k} &= \left(\tfrac{u_{i,j+1,k}-u_{i,j,k}}{h}\right)^2 + \left(\tfrac{u_{i+1,j+1,k}-u_{i-1,j+1,k}+u_{i+1,j,k}-u_{i-1,j,k}}{h}\right)^2 \\
&+ \left(\tfrac{u_{i,j+1,k+1}-u_{i,j+1,k-1}+u_{i,j,k+1}-u_{i,j,k-1}}{h}\right)^2
\end{aligned}$$

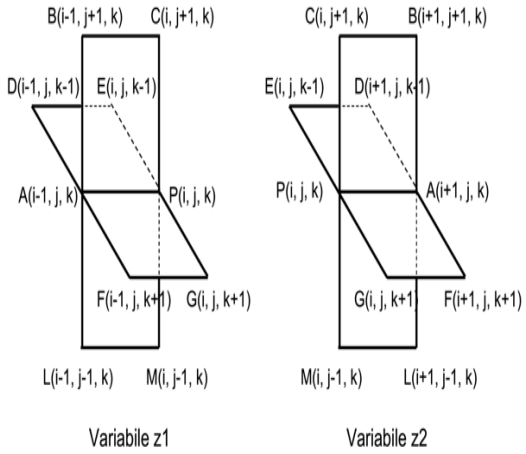

Figura 2.9: *Nella figura sono rappresentati i punti della griglia in cui valutare la funzione u^n per il calcolo delle variabili $z1$ e $z2$ in $P(i,j,k)$.*

$$
\begin{aligned}
z5^b_{i.j.k} &= \left(\frac{u_{i,j,k}-u_{i,j,k-1}}{h}\right)^2 + \left(\frac{u_{i+1,j,k-1}-u_{i-1,j,k-1}+u_{i+1,j,k}-u_{i-1,j,k}}{h}\right)^2 \\
&+ \left(\frac{u_{i,j+1,k-1}-u_{i,j-1,k-1}+u_{i,j+1,k}-u_{i,j-1,k}}{h}\right)^2 \\
z6^t_{i,j,k} &= \left(\frac{u_{i,j,k+1}-u_{i,j,k}}{h}\right)^2 + \left(\frac{u_{i,j+1,k+1}-u_{i,j-1,k+1}+u_{i,j+1,k}-u_{i,j-1,k}}{h}\right)^2 \\
&+ \left(\frac{u_{i+1,j,k+1}-u_{i-1,j,k+1}+u_{i+1,j,k}-u_{i-1,j,k}}{h}\right)^2 \\
z7_{i,j,k} &= \tfrac{1}{6}\textstyle\sum_{i=1}^{6} zi_{i,j,k}.
\end{aligned}
$$

Nelle Figure 2.9, 2.10 e 2.11 sono indicati i punti della griglia in cui viene valutata la funzione u^n per il calcolo nel punto $P(i,j,k)$ rispettivamente delle variabili $z1^w_{i.j.k}$ e $z2^e_{i,j,k}$, $z3^w_{i.j.k}$ e $z4^e_{i,j,k}$, $z5^b_{i.j.k}$ e $z6^t_{i,j,k}$. Si noti che $z7_{i,j,k}$ è una media delle zi per $i = 1, \ldots, 6$.

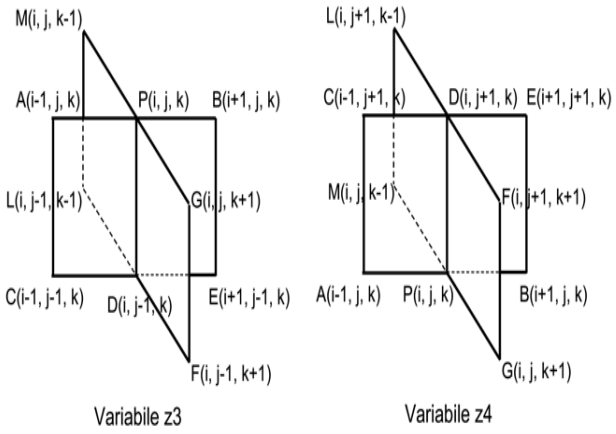

Figura 2.10: *Nella figura sono rappresentati i punti della griglia in cui valutare la funzione u^n per il calcolo delle variabili $z3$ e $z4$ in $P(i, j, k)$.*

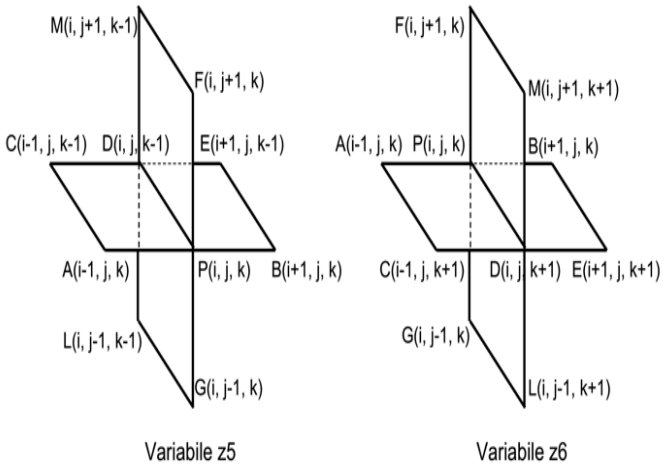

Figura 2.11: *Nella figura sono rappresentati i punti della griglia in cui valutare la funzione u^n per il calcolo delle variabili $z5$ e $z6$ in $P(i, j, k)$.*

Fissato $P(i,j,k)$ definiamo la costante

$$app = \tau * \sqrt{\epsilon + z7}\ ,$$

ed i coefficienti che rappresentano i contributi di a^n_{pq} al variare del volume di controllo q (aw, ae, as, an, ab, at) ed in P stesso (ap):

$$\begin{array}{rclcrcl} aw^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z1}} & e & ae^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z2}} \\ as^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z3}} & e & an^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z4}} \\ ab^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z5}} & e & at^n_{i,j,k} & = & app * \frac{1}{\sqrt{\epsilon+z6}} \end{array}$$

$$ap^n_{i,j,k} = h^2 + aw^n_{i,j,k} + ae^n_{i,j,k} + as^n_{i,j,k} + an^n_{i,j,k} + ab^n_{i,j,k} + at^n_{i,j,k}$$

La (2.20) può essere, quindi, riscritta come:

$$\begin{aligned} &ap_{i,j,k}u^n_{i,j,k} - aw_{i,j,k}u^n_{i-1,j,k} - ae_{i,j,k}u^n_{i+1,j,k} - as_{i,j,k}u^n_{i,j-1,k} \\ &-an_{i,j,k}u^n_{i,j+1,k} - ab_{i,j,k}u^n_{i,j,k-1} - at_{i,j,k}u^n_{i,j,k+1} = r_{i,j,k} \end{aligned} \tag{2.22}$$

con i coefficienti dipendenti dagli zi e quindi da u.
Considerata l'espressione in forma matriciale della (2.22):

$$F(u(x,t)) = Au(x,t) - b = 0$$

La matrice $A \in \Re^2$ è piena ed ha dimensione $(l_1 \cdot l_2 \cdot l_3) \times 7$; le righe sono $l_1 \cdot l_2 \cdot l_3$ in quanto c'è una corrispondenza uno ad uno tra i punti della griglia e le righe della matrice A; le colonne sono 7 in quanto c'è una corrispondenza uno ad uno tra i punti dell'insieme $C_p = \{w, e, s, n, b, t\} \cup p$ (Figura 2.3) e le colonne della matrice A, costituite dai vettori

$$\{(ap_{i,j,k}, -aw_{i,j,k}, -ae_{i,j,k}, -as_{i,j,k}, -an_{i,j,k}, -ab_{i,j,k}, -at_{i,j,k})\}_{(i,j,k)}.$$

Il vettore $u(x,t)$, al variare di $P(i,j,k)$ ha componenti

$$(u^n_{i,j,k}, u^n_{i-1,j,k}, u^n_{i+1,j,k}, u^n_{i,j-1,k}, u^n_{i,j+1,k}, u^n_{i,j,k-1}, u^n_{i,j,k+1})^T,$$

mentre gli elementi di b coincidono con $\{r_{i,j,k}\}_{(i,j,k)}$.

Capitolo 3

Il metodo numerico

Consideriamo la forma matriciale del sistema di equazioni non lineare (2.22)

$$F(u(x,t)) = Au(x,t) - b = 0 \tag{3.1}$$

Per la risoluzione del sistema (3.1) è stato usato il metodo **JFNK** (**J**acobian-**F**ree **N**ewton-**K**rylov)[30].

Il metodo **JFNK** è un metodo iterativo, articolato usualmente in quattro livelli. Il primo livello base è definito dall'approssimazione lineare del metodo di Newton; il secondo è quello relativo alla risoluzione del sistema lineare con un metodo iterativo. All'interno è presente spesso un precondizionatore. Esternamente al ciclo di Newton è utilizzato un metodo di globalizzazione per la scelta dell'approssimazione iniziale che garantisca la convergenza alla soluzione (IV livello). Nella Procedura 3.1 sono schematizzati i quattro livelli del **JFNK**.

```
begin - JFNK
  ⋮
  % Loop di Globalizzazione (IV livello)
   do {
      % Loop del metodo di Newton (I livello)
       do {
         % Loop di Krylov (II livello)
          do {
            % Loop del Precondizionatore (III livello)
             do {
                   ⋮
                } while (Arresto Precondizionatore)
              } while (Arresto Krylov)
            } while (Arresto Newton)
         } while (Arresto Globalizzazione)
  ⋮
end
```

*Procedura 3.1 - Schema del metodo **JFNK**.*

3.1 Richiami sul metodo di Newton.

Assegnato il sistema di equazioni non lineari:

$$F(u(x,t)) = 0 \ \ in\ \Omega, \tag{3.2}$$

consideriamo il suo sviluppo in serie di Taylor di punto iniziale z e valutato in w:

$$F(w) = F(z) + F'(z)(w - z) + termini\ ordine\ superiore$$

Trascurando i termini dello sviluppo in serie di Taylor di ordine superiore al primo si ottiene:

$$F(w) = F(z) + J(z) \cdot a = 0 \tag{3.3}$$

avendo indicato con $a = w - z$ e $J(z) = F'(z)$. $J(z)$ è la matrice jacobiana della $F(z)$:

$$J(i,j) = \frac{\partial F_i(u)}{\partial u_j} \quad \forall i,j$$

Dalla (3.3) si ha:

$$\begin{cases} J(z) \cdot a = -F(z) \\ w = z + a \end{cases} \tag{3.4}$$

Sia $k = 0, 1, \ldots$. Posto

$$\begin{cases} z &= u^{(k)} \\ w &= u^{(k+1)} \\ a &= \delta u^{(k)} \end{cases}$$

ad ogni passo k il generico vettore $u^{(k+1)}$ viene calcolato a partire

dal precedente $u^{(k)}$ risolvendo il sistema:

$$\begin{cases} J(u^k)\delta u^k = -F(u^k) \\ u^{k+1} = u^k + \delta u^k \end{cases} \tag{3.5}$$

con u^0 assegnata.
Il criterio di arresto si basa sulla riduzione del residuo

$$\frac{\|F(u^k)\|}{\|F(u^0)\|} < tol_{res}$$

e/o su un aggiornamento della soluzione

$$\left\| \frac{\delta u^k}{u^k} \right\| < tol_{update}$$

con tol_{res} e tol_{update} quantità pre definire.

3.2 Richiami sui metodi di Krylov.

I metodi basati sui sottospazi di Krylov sono utilizzati per la risoluzione di sistemi lineari di grandi dimensioni. Questi metodi sono di proiezione (Galerkin) o proiezione generalizzata (Petrov-Galerkin) [44] per la risoluzione di $Ax = b$ usando il sottorpazio di Krylov

$$K_j = span\left(r_0, Ar_0, A^2 r_0, \ldots, A^{j-1} r_0\right)$$

dove $r_0 = b - Ax_0$. Richiedono solo prodotti matrice-vettore e questa è la chiave principale per il loro ampio utilizzo.
Una grande varietà di metodi iterativi utilizzano la tassonomia di Krylov [30, 44] ed una classificazione importante la si è avuta in relazione all'applicabilità a sistemi non simmetrici (es. la matrice Jacobiana non è simmetrica). Un ulteriore criterio di distinzione è in re-

lazione alla procedura di ortogonalizzazione di Arnoldi, la quale genera una base ortonormale del sottospazio di Krylov, e la procedura di bi-ortogonalizzazione di Lanczos, la quale genera, invece, basi non ortogonali. Il metodo con base di Arnoldi più utilizzato è il **GMRES** (**G**eneralize **M**inimal **RES**idual), mentre metodi su basi di Lanczos sono il **BiCGSTAB** (**Bi-C**oniugate **G**radient **STAB**ilized) ed il **TFQMR** (**T**ranspose-free **Q**uasi **M**inimal **R**esidual).

3.3 Il metodo JFNK

Le origini del metodo **JFNK** si possono far risalire alle publicazioni relative alla risoluzione delle ODE [7, 23] e delle PDE [8, 13]. La motivazione principale appare essere la capacità di eseguire iterazioni di Newton senza generare l'intera matrice Jacobiana, per questo motivo può essere inserito nella classe dei metodi *"quasi Newton"*. Inoltre, i metodi **JFNK** sono detti *Jacobian-free* perchè, in analogia ai metodi di tipo *matrix-free*, ciò che viene calcolato senza far uso esplicito della matrice Jacobiana è il prodotto di questa con un generico vettore. L'idea dei metodi **JFNK** si basa sul risultato seguente:

Proposizione 1:
Assegnata una funzione reale a valori reali $F(u) \in C^1(\Re^d)$*, la matrice Jacobiana,* $J = F'(u)$*, ed un vettore reale* v*, il prodotto* Jv *può essere approssimato dallo sviluppo in serie di Taylor del primo ordine di* F*, ovvero*[30]:

$$J(u) \cdot v = \frac{F(u + \epsilon v) - F(u)}{\epsilon} + O(\epsilon) \tag{3.6}$$

con $\epsilon > 0$.

Dimostrazione:
Senza ledere la generalità fissiamo lo spazio $\Re^2$. *Se* $F = (F_1, F_2)$ *allora*

$$F(u) = 0 \Longleftrightarrow \begin{cases} F_1(u_1, u_2) = 0 \\ F_2(u_1, u_2) = 0 \ , \end{cases}$$

Consideriamo il vettore $v = (v_1, v_2)$ *e la matrice Jacobiana* J

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} \end{bmatrix}.$$

Si ha:

$$\frac{F(u+\epsilon v) - F(u)}{\epsilon} = \begin{pmatrix} \frac{F_1(u_1+\epsilon v_1, u_2+\epsilon v_2) - F_1(u_1,u_2)}{\epsilon} \\ \frac{F_2(u_1+\epsilon v_1, u_2+\epsilon v_2) - F_2(u_1,u_2)}{\epsilon} \end{pmatrix}$$

Approssimando $F(u+\epsilon v)$ *con lo sviluppo di* F *in serie di Taylor del primo ordine relativo ad* u, *otteniamo*

$$\frac{F(u+\epsilon v) - F(u)}{\epsilon} = \begin{pmatrix} \frac{F_1(u_1,u_2) + \epsilon v_1 \frac{\partial F_1}{\partial u_1} + \epsilon v_2 \frac{\partial F_1}{\partial u_2} - F_1(u_1,u_2)}{\epsilon} \\ \frac{F_2(u_1,u_2) + \epsilon v_1 \frac{\partial F_2}{\partial u_1} + \epsilon v_2 \frac{\partial F_2}{\partial u_2} - F_2(u_1,u_2)}{\epsilon} \end{pmatrix} + O(\epsilon)$$

da cui a meno di un infinitesimo di ordine ϵ

$$\frac{F(u+\epsilon v) - F(u)}{\epsilon} = \begin{pmatrix} v_1 \frac{\partial F_1}{\partial u_1} + v_2 \frac{\partial F_1}{\partial u_2} \\ v_1 \frac{\partial F_2}{\partial u_1} + v_2 \frac{\partial F_2}{\partial u_2} \end{pmatrix} = Jv$$

$\triangle$

Quindi, l'errore che si commette nell'approssimazione di Jv è proporzionale ad ϵ [30]. L'approccio *matrix-free* ha molti vantaggi, il

più interessante è permettere la convergenza non lineare di un "quasi Newton" non richiedendo il calcolo esplicito di tutti gli elementi della matrice Jacobiana.

È immediato determinare l'approssimazione del secondo ordine troncando lo sviluppo di Taylor al secondo ordine:

$$Jv = \frac{F(u + \varepsilon v) - F(u - \varepsilon v)}{\varepsilon} + O(\epsilon^2).$$

La convergenza per il **JFNK** è stata dimostrata in [6, 9], le condizioni sono fissate al variare della dimensione di ε che garantisce la convergenza locale. Condizioni sulla convergenza globale vengono studiate nel [8]. Due situazioni che causano problemi alla convergenza del **JFNK** sono la struttura non lineare della soluzione e la discontinuità nella funzione F.

3.3.1 Fase di globalizzazione.

Il metodo utilizzato in questo lavoro è la *line search*, il quale calcola un valore s (≤ 1) utilizzato per l'aggiornamento della soluzione al passo corrente:

$$u^{k+1} = u^k + s\ \delta u^k$$

Il semplice test utilizzato per determinare il parametro s è la richiesta di riduzione del residuo:

$$F(u^k + s\ \delta u^k) < F(u^k)$$

3.3.2 Introduzione del precondizionatore

Usando un precondizionatore destro si deve risolvere il sistema

$$(JP^{-1})(P\delta u) = -F(u) \tag{3.7}$$

La risoluzione della (3.7) consta di due passi. Nel primo si risolve rispetto a w il sistema:

$$(JP^{-1})w = -F(u)$$

con $w = (P\delta u)$, per poi, nel secondo calcolare:

$$\delta u = P^{-1}w$$

Riprendendo la **Proposizione 1**, con l'introduzione del precondizionatore destro otteniamo l'approssimazione seguente:

$$J(u) \cdot P^{-1} \cdot v = \frac{F(u + \epsilon P^{-1}v) - F(u)}{\epsilon} + O(\epsilon) \tag{3.8}$$

Nella Procedura 3.2 è illustrato uno schema dell'algoritmo implementato.

begin
⋮
$u(x,0) = u_0$
for $i = 1$ **to** $nscale$ **do**
% Loop risolutivo del metodo di Newton
$k = 1$
do {
% Costruzione del sistema lineare
$J(u^k) \cdot \delta u^k = -F(u^k)$
% Loop risolutivo del sistema lineare
% mediante GMRES con Precondizionatore
do {
% Determinazione di δu^k
} **while** (criterio di arresto)
% Aggiornamento della funzione $u(x,t)$
% mediante il metodo della line search
$u^{k+1} = u^k + s\delta u^k$
} **while** (criterio di arresto)
endfor
⋮
end

Procedura 3.2 - Schema dell'algoritmo implementato.

3.3.3 Alcune scelte per il fattore ε

Analizziamo alcune possibili scelte del parametro ε presente nella (3.6).
Iniziamo con l'osservare che se ε assume un valore *"troppo grande"* ($\varepsilon \gg 0$), il valore calcolato è un'approssimazione poco accurata, se, viceversa, ε è molto *"piccolo"* ($\varepsilon \to 0$), il risultato numerico in questo caso si può amplificare notevolmente, effetto dell'errore di roundoff. In [13] ε è stato posto uguale ad un valore sufficientemente grande: la radice quadrata dell'epsilon macchina (ϵ_{mach}) (quindi con $\varepsilon = 10^{-3}$).

Altri approcci per la scelta di ε non assegnano un valore costante, ma effettuano calcoli all'interno dell'algoritmo stesso, usando i valori dello Jacobiano o colonne dello stesso. Una semplice scelta può essere [30]:

$$\varepsilon = \frac{1}{n\|v\|_2} \sum_{i=1}^{n} b|u_i| + b$$

dove n rappresenta la dimensione del sistema lineare e b è una costante il cui valore è dell'ordine di grandezza della radice quadrata dell'errore di roundoff della macchina (tipicamente è dell'ordine di 10^{-6} per una doppia precisione a 64 bit). Questo approccio produce l'ε *medio* che si otterrebbe se ogni singolo elemento dello Jacobiano fosse calcolato come:

$$J_{i,j} = \frac{F_i(u + \varepsilon_j e_j) - F_i(u)}{\varepsilon_j}$$

con

$$\varepsilon_j = bu_j + b$$

ed il vettore e_j costituito da componenti tutte nulle eccetto la j_{esima}

che è uguale ad 1.

Un'altro approccio più complesso, proposto da Brown e Saad [8], ha il suo nucleo nel calcolo delle colonne dello Jacobiano [18] ed è dato da

$$\varepsilon = \frac{b}{\|v\|_2}\mathbf{max}\{|u^T v|, \mathbf{typ}\ u|v|\}\mathbf{sign}(u^T v)$$

dove la funzione "**tpy** u" indica la *"tipica dimensione"* di u.
È importante notare che la scelta di $b = \sqrt{\varepsilon_{mach}}$ porta ad assumere che $F(u)$ possa essere calcolata con la precisione dell'ε macchina; viceversa se si conosce la possibilità di valutare la funzione $F(u)$ con la precisione dell'ε macchina assegnare a b il valore $\sqrt{\varepsilon_{mach}}$ risulta una buona scelta.

Quando è noto che c'è una limitazione sulla precisione della valutazione della $F(u)$ è utilizzata un'altra formula per la valutazione di ε [30] (approccio di Wolker e Pernice):

$$\varepsilon = \frac{\sqrt{(1 + \|u\|)\varepsilon_{mach}}}{\|v\|}$$

Lo scopo principale di ciscun approccio è quello di ridurre il numero di iterazioni richieste dal Newton ottenendo la maggiore accuratezza possibile nell'approssimazione del prodotto Jv.

Capitolo 4

L'ambiente di sviluppo dell'algoritmo: PETSc

PETSc (**P**ortable, **E**xstensible **T**oolkit for **S**cientific **C**omputation) è una libreria che raccoglie strutture dati e software che rappresentano i *"building blocks"* per l'implementazione di una larga scala di applicazioni in ambiente di calcolo ad alte prestazioni modellate da equazioni differenziali alle derivate parziali. **PETSc** è una libreria flessibile, i suoi moduli, che possono essere utilizzati in codici scritti in *Fortran*, *C* o C^{++}, sono stati sviluppati secondo la logica della programmazione ad oggetti. La libreria, inoltre, ha una struttura gerarchica (Figura 4.1) che indica come la scelta del solutore del problema dipenda dal livello di astrazione. Nella Figura 4.1, in particolare, sono illustrati:

- i nuclei di base computazionali (BLAS e LAPACK per l'algebra lineare) e di Message Passing (MPI) utilizzati da **PETSc**;
- la possibilità di scegliere il livello di trasparenza desiderato nella risoluzione del problema: si possono utilizzare solutori *built-in*

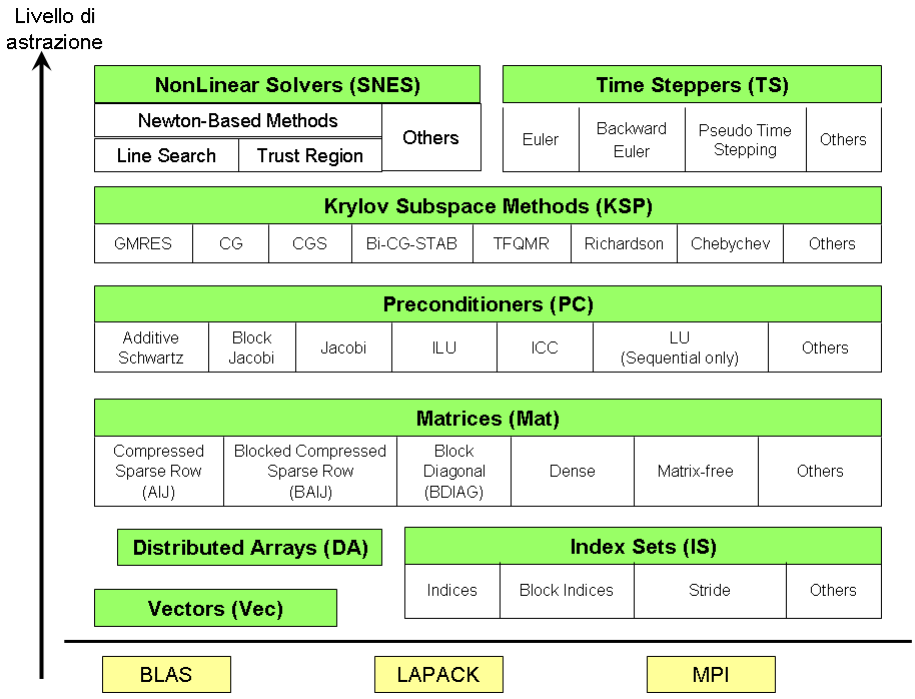

Figura 4.1: Struttura Gerarchica di **PETSc**

per equazioni differenziali o intervenire ad un livello più basso di astrazione implementando la discretizzazione e risolvendo i sistemi con moduli della libreria quali ad esempio: LIMPACK, MINPACK, SPARSEKIT2; **PETSc**, inoltre, mette a disposizione strutture dati utili nella definizione delle griglie utilizzate per la discretizzazione delle PDE, rendendo trasparente all'utente la gestione degli eventuali bordi di *overlap*;

- la disponibilità di solutori lineari iterativi basati su metodi di proiezione di Krylov e relativi precondizionatori, quali ad esempio GMRES e CG che possono essere precondizionati mediante Additive Schwartz, Block Jacobi, ILU, ICC;

- la disponibilità di solutori non lineari iterativi basati sul metodo di Newton e relativi metodi di globalizzazione, quali la *line search* e la *trust region*.

È inoltre possibile l'integrazione con alcune librerie esterne, quali:

- ADIC-ADIFOR (Calcolo automatizzato delle matrici sparse Jacobiane);
- AMG (Metodi multigrid);
- BlockSolve95 (Precondizionatori ICC(0) e ILU(0) paralleli);
- DSCPACK (Cholesky sparso);
- ESSL (Libreria IBM per le fattorizzazioni LU sparse);
- EUCLId (Fattorizzazione ILU(k));
- LUSOL (Fattorizzazione LU sparsa);
- Mathematica;
- Matlab;
- SPAI (Precondizionatori basati sull'approssimazione sparsa dell'inversa);
- SuperLU e SuperLU_Dist (Solutori diretti sparsi, sequenziali e per calcolatori paralleli a Memoria Distribuita);
- ParPre (Libreria di precondizionatori implementati in ambiente parallelo, basati su metodi di decomposizione e metodi multilivello).

La possibilità di integrare **PETSc** con numerose librerie scientifiche permette di effettuare agevolmente esperimenti e confronti tra i vari metodi di risoluzione.

PETSc per ogni elemento che deve essere considerato nel problema assegnato, ad esempio un vettore e una matrice nel prodotto matrice-vettore, costruisce il corrispondente oggetto **Vec** e **Mat**[1]. A livello più elevato di astrazione, e quindi all'aumentare della complessità del problema assegnato, **PETSc** si comporta in modo analogo; ad esempio:

- per la risoluzione di un sistema lineare costruisce l'oggetto **KSP**, che può includere oppure no l'oggetto precondizionatore **PC**, a seconda che il metodo risolutivo scelto lo si voglia precondizionare o meno;
- per la risoluzione di sistemi non linari costruisce l'oggetto **SNES** che include in sé un oggetto **KSP** per la risoluzione del sistema linearizzato con il metodo che caratterizza lo **SNES**.

4.1 L'algoritmo implementato in PETSc

Nella Figura (4.2) sono illustarti i principali moduli per l'implementazione dell'algoritmo parallelo relativo alla segmentazione e al denoising di immagini 3D in ambiente **PETSc**; in particolare sono indicati in riquadri blu i moduli che l'utente deve implementare (routine principale, di inizializzazione, di valutazione della funzione e di raccolta dei dati di output) e in riquadri bianchi quelli che offre **PETSc** (risolutore non lineare, risolutore lineare, precondizionatore, valutazione dello Jacobiano).

Nella Procedura 3 è illustrato lo schema *flow chart* in cui sono indicate le principali routines di **PETSc** richiamate:

[1] Ciascun oggetto **PETSc** è stato definito in relazione all'operazione che deve eseguire e non ai dati che deve contenere. Ad esempio un vettore **VEC** non è considerato un array di valori, ma un oggetto astratto su cui sono definite operazioni di addizione e moltiplicazione scalari.

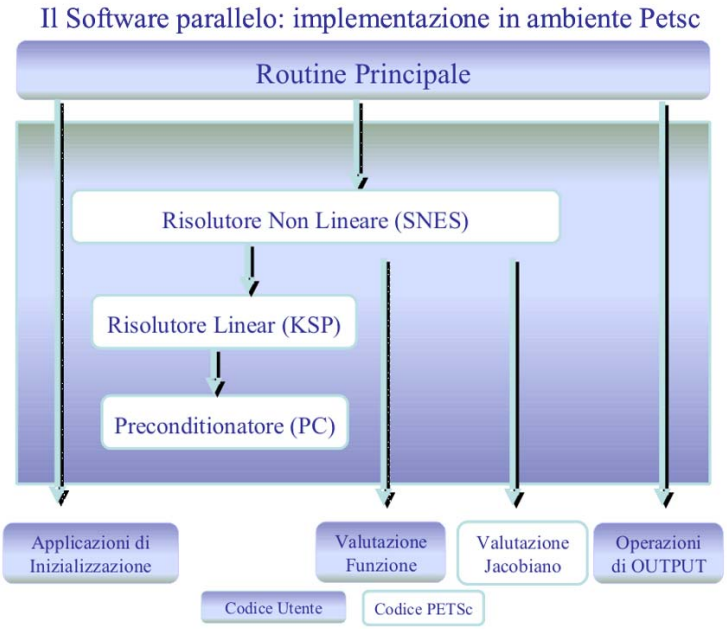

Figura 4.2: In figura è illustrarto lo schema dell'algoritmo implementato in ambiente **PETSc**

- per l'inizializzazione e la chiusura dell'ambiente: $PetscInizialize(\dots)$, $PetscFinalize(\dots)$;
- per la creazione della griglia distribuita 3D su cui definire gli oggetti Vec e Mat: $DACreate3d(\dots)$;
- per la creazione dell'oggetto che caratterizza il risolutore non lineare: $SNESCreate(\dots)$;
- per l'inizializzazione sulla griglia distribuita della funzione rappresentativa del sistema non lineare: $DASetLocalFunction(\dots)$;
- per l'inizializzazione sulla griglia distribuita dello Jacobiano della funzione rappresentativa del sistema: $SNESSetJacobian(\dots)$;

- per l'assegnazione dei valori iniziali: $FromInitialGuess(\dots)$;
- per la risoluzione del sistema non lineare: $SNESSolve(\dots)$;
- per la distruzione degli oggetti PETSc: $DADestroy(\dots)$ e $SNESDestroy(\dots)$.

```
begin
   ⋮
   % Inizializzazione ambiente PETSc
    PetscInizialize(...)
   % Creazione della griglia distribuita 3D
    ierr =DACreate3d(...)
   % Creazione dell'oggetto SNES
   % (risolutore non lineare)
    ierr=SNESCreate(...)
   % Settaggio della funzione F
    ierr=DASetLocalFunction(...)
   % Settaggio dello Jacobiano
    ierr=SNESSetJacobian(...)
   % Assegnazione dei valori iniziali
    ierr=FromInitialGuess(...)
   % Risoluzione del sistema non lineare
    ierr=SNESSolve(...)
```

Procedura 4.1 - Schema d'implementazione in ambiente PETSc - continua

```
    % Distruzione della griglia DA
     ierr=DADestroy(...)
    % Distruzione dello SNES
     ierr=SNESDestroy(...)
    % Chiusura dell'ambiente PETSc
     PetscFinalize(...)
    ⋮
end
```

Procedura 4.1 - Schema d'implementazione in ambiente PETSc - fine

4.1.1 Gli oggetti DA definiti in PETSc

Gli oggetti DA (**D**istributed **A**rray) sono intesi come oggetti PETSc in grado di comunicare tra loro in modo ottimizzato delle informazioni e non come semplici array in grado di parallelizzare dati memorizzati in array. Negli oggetti DA:

- si definisce un particolare schema di numerazione dei dati distribuiti tra i processori, diverso da quello naturale definito sul dato globale (Figura 4.3);
- si definiscono indici globali e locali su di uno stesso oggetto (Figura 4.4);
- si definiscono vettori globali e locali;
- nei Vec globali sono definiti i punti di *"ghost"* (Figura 4.5), fondamentali, ad esempio, nella risoluzione di una PDE in pa-

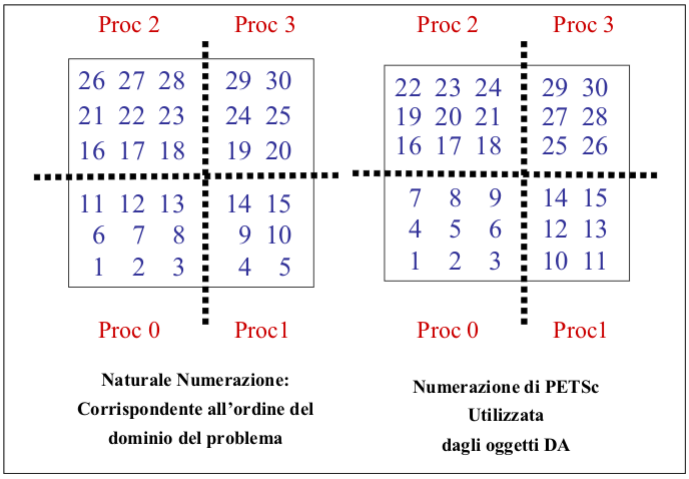

Figura 4.3: In figura sono messi a confronto lo schema di numerazione naturale dei dati distribuiti tra 4 processori (Proc0, Proc1, Proc2 e Proc3) e il corrispondente utilizzato da **PETSc** negli oggetti di tipo DA.

rallelo; in questo caso si ha la neccessità di valutare una funzione $f(x)$ e ad ogni processore, per farlo, occorre sia la propria porzione del vettore x sia quella posseduta da altri;

- si effettuano facilmente operazioni di visualizzazione e aggiornamento dell'oggetto DA globale.

4.1.2 Calcolo del fattore ϵ in PETSc

Nell'ambito dello SNES definito in **PETSc** il fattore ϵ si può calcolare seguendo due metodi distinti:

1. il metodo descritto da Brown e Saad;
2. il metodo descritto da Walker e Pernice.

Nel metodo descritto da Brown e Saad il fattore ϵ è calcolato usando l'espressione seguente:

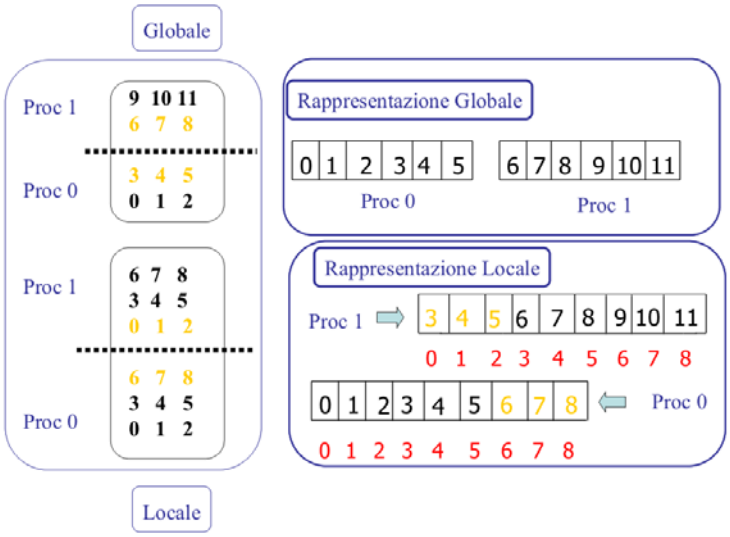

Figura 4.4: In figura a sinistra è illustrata l'indicizzazione globale e locale di un Vec di tipo DA distribuito tra 2 processori (Proc0 e Proc1); a destra la rappresentazione globale e locale dei suoi elementi. Si può notare come nella rappresentazione locale Proc0 conosca i primi tre elementi di Proc1, il quale, a sua volta, conosce gli ultimi tre elementi di Proc0.

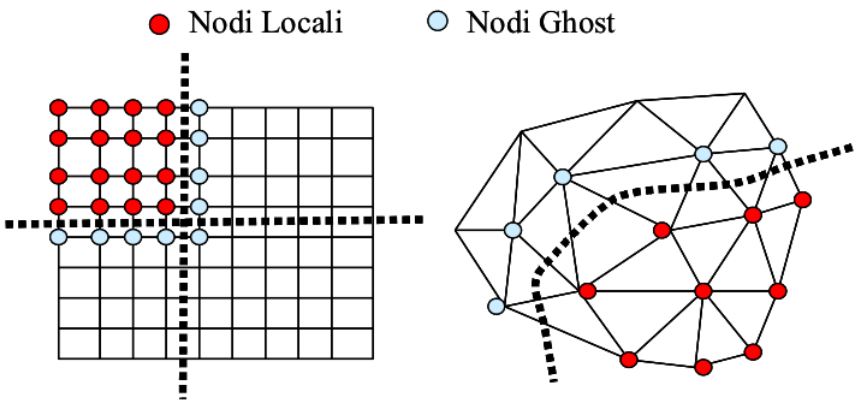

Figura 4.5: In figura è illustrato un esempio di punti locali (in rosso) e di punti di ghost (in azzurro) su due griglie distinte; il tratteggio indica la linea di separazione tra gli elementi appartenenti a processori distinti.

$$\epsilon = \begin{cases} e_{rel} * u^T v / \|v\|_2^2 & se\ |u^T v| > u_{min} * \|v\|_1 \\ e_{rel} * u_{min} * sign(u^T v) * \|v\|_1 / \|v\|_2^2 & altrimenti \end{cases}$$

Per default $u_{min} = 10^{-6}$ ed $e_{rel} = 10^{-8}$. Nei test eseguiti ho considerato questo metodo per il calcolo del fattore ϵ.

Nel metodo descritto da Walker e Pernice il fattore ϵ è calcolato usando l'espressione seguente:

$$\epsilon = \frac{\sqrt{1 + \|u\|} e_{rel}}{\|v\|}$$

4.1.3 Approssimazione dello Jacobiano con le differenze finite in PETSc

Nell'ambito dello SNES definito in **PETSc** si può calcolare la matrice Jacobiana di una funzione $F(u)$ utilizzando le differenze finite a partire da $F(u)$:

$$J_{:i} \approx \frac{F(u + \epsilon * dx_i) - F(u)}{\epsilon}$$

con ϵ calcolata nel modo seguente:

$$\epsilon = \begin{cases} e_{rel} * u_i & se\ |u_i| > u_{min} \\ e_{rel} * u_{min} * sign(u_i) & altrimenti \end{cases}$$

Per default $u_{min} = 10^{-8}$ ed $e_{rel} = 10^{-8}$.

Capitolo 5

Il software parallelo e sua applicazione su immagini mediche

Molti problemi di immagini richiedono una quantità elevata di dati. L'ordine di grandezza della complessità di spazio e tempo per l'algoritmo implementato della segmentazione/denoising è il seguente:

$$T_n = O\left(Ts \cdot \{NL \cdot [L \cdot (b1 \cdot n^2 + b2 \cdot n) + b2 \cdot n + 7 \cdot n^2]\}\right)$$
$$S_n = O\left(n^2\right)$$

avendo indicato con:

- Ts - il numero di passi temporali;
- NL - il numero di passi non lineari;
- L - il numero di passi lineari;
- $b1 \cdot n^2$ - l'ordine della complessità del prodotto matrice per vettore nel solutore lineare;

- $b2 \cdot n$ - l'ordine della complessità sia della risoluzione del sistema per il precondizionatore che della ILU(0);
- $7 \cdot n^2$ - l'ordine della complessità delle operazioni necessarie per la costruzione della matrice del sistema in forma compatta.

Questo significa che se si considera un immagine *"piccola"*, ad esempio, di dimensione $n = 128 \times 100 \times 83 = O(10^6)$, lo jacobiano $J(A)$ ha dimensione $m = n \times n = O(10^{12})$, e quindi la complessità computazionale è:

$$T_n = O(Ts \cdot NL \cdot L \cdot n^2) = O(Ts \cdot NL \cdot L \cdot 10^{12}) \text{ flop}$$

$$S_n = O(10^{12})$$

Se si tiene conto che il prodotto $Ts \cdot NL \cdot L$ è almeno dell'ordine di 10^4, l'ordine di grandezza di T_n è di almeno $O(10^{16})$. Se consideriamo un PC che esegue 1 $Gflop$ di operazioni f.p. al secondo per la risoluzione del problema sono necessari:

$$\frac{10^{16}\ flop}{1\ Gfolp/sec} = \frac{10^7\ Gflop}{1\ Gfolp/sec} = 10^7 sec \approx 4\ mesi.$$

Evidentemente è un tempo di risposta inaccettabile per qualsiasi applicazione concreta.

5.1 Introduzione del parallelismo

La strategia utilizzata per introdurre il parallelismo segue un approccio basato sulla decomposizione del dominio, nel caso specifico la strategia utilizzata è denominata *"Slice Partitioning"*, in quanto il dominio 3D è partizionato lungo un'unica dimensione (Figura 5.1).

Sia $\Omega \in \Re^3$ il dominio rettangolare dell'immagine, di dimensione $n_1 \times n_2 \times n_3$. Il dominio è stato suddiviso solo lungo la terza di-

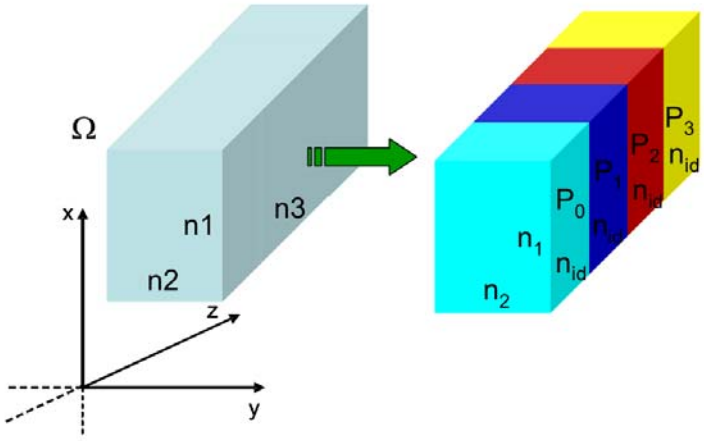

Figura 5.1: In figura è illustrata la distribuzione di un dominio Ω (in azzurro) di dimensione $n_1 \times n_2 \times n_3$ tra 4 processori: P_0, P_1, P_2 e P_3. Ciascun processore contiene una sezione di Ω di dimensione $n_1 \times n_2 \times n_{id}$.

mensione, quindi se p è il numero di processori e l'identificativo id di ciascuno è un intero compreso tra 0 e $(p-1)$, Ω verrà partizionato nell'insieme di voxel $\{Vloc_i\}_{i=0,\ldots,p-1}$, ciascuno di dimensione $n_1 \times n_2 \times n_{id}$, con

$$n_{id} = \begin{cases} n_3/p + 1 & se \;\; id < mod(n_3, p) \\ n_3/p & altrimenti \end{cases} \tag{5.1}$$

Con tale distribuzione si ha che ogni processore effettua calcoli su di un volume $Vloc_i$ di dimensione $n_1 \times n_2 \times n_{id}$ mentre effettua comunicazioni dei dati appartenenti alle 2 superfici di separazione S_{loc} dei sotto domini, ciascuna delle quali ha dimensione $n_1 \times n_2$ [1].

[1] I processori il cui identificativo è 0 e $p-1$ hanno una sola superficie di separazione e quindi il numero totale dei dati per la comunicazione è $n_1 \times n_2$.

Considerando l'overhead totale di comunicazione:

$$OC_p = \frac{T_{com}}{T_{calc}}$$

segue che:

$$OC_p = \frac{T_{com}}{T_{calc}} = \frac{S_{loc}}{Vloc_i} = \frac{2}{n_{id}}.$$

Quindi ogni 2 comunicazioni vengono effettuate n_{id} operazioni f.p., tale decomposizione scelta porta ad una significativa riduzione dell'overhead di comunicazione. D'altra parte, però, considerando la (5.1) si hai anche:

$$OC_p = \frac{T_{com}}{T_{calc}} = \frac{2p}{n_3}$$

ovvero, fissata la dimensione del problema, e quindi n_3, facendo crescere il numero dei processori p aumenta proporzionalmente anche OC_p ovvero aumenta il peso del tempo di comunicazione sul tempo di calcolo, con degradazione delle prestazioni dell'algoritmo. In altre parole ci dobbiamo aspettare che, fissato n_3, esista un valore *"ottimale"* di p al di là del quale le prestazioni dell'algoritmo parallelo possono degradare. Del resto questa situazione era prevedibile considerando che

$$E_p = \frac{1}{1 + \frac{O_h}{T_1}} \tag{5.2}$$

Se O_h aumenta l'efficienza degrada.

5.2 Le Immagini Ecografiche

L'ecografia è una tecnica di diagnosi medica per immagini che consente di rappresentare gli organi del nostro corpo utilizzando la ri-

sposta dei tessuti alle onde sonore ad alta frequenza (ultrasuoni non udibili dall'orecchio umano) che attraversano i tessuti stessi. Il principio su cui si basa l'ecografia è lo stesso utilizzato dal sonar delle navi per localizzare i sottomarini. La sonda posta sul paziente, ad esempio sull'addome, invia impulsi di onde sonore nel corpo. Queste onde sonore in parte vengono riflesse dalla parete addominale creando echi, in parte arrivano all'organo che si vuole visualizzare, il quale le riflette: tali onde di ritorno sono trasformate sul monitor dell'ecografo in immagini. La sonda, é composta di cristalli che hanno la funzione di inviare nonché di ricevere gli echi di ritorno. Il computer dell'ecografo è in grado di calcolare in tempo reale, impensabili in passato, sia l'intervallo trascorso tra l'ultrasuono inviato e l'eco ricevuto, sia l'intensità con cui tale eco viene riflesso. Da tale calcolo, eseguito punto per punto dello schermo in tempo reale, l'ecografo è in grado di ricostruire l'immagine (Figura 5.2): l'intervallo di tempo determinerà le coordinate dell'interfaccia incontrata e l'intensità indicherà il livello di grigio da rendere sullo schermo. Ogni pixel quindi sarà calcolato in questo modo, molte volte al secondo. L'immagine che si ottiene mostra quindi in maniera bidimensionale sullo schermo una sezione della struttura esaminata, consentendo anche di distinguere eventuali movimenti.

Figura 5.2: In figura è illustrato il processo di generazione di un immagine ecografica.

5.3 Risultati

I risultati ottenuti sono illustrati utilizzando due tipi di immagini test, entrambe ottenute con una sonda ecografica: le prime sono il risultato della scansione di oggetti noti, quali dadi, gomme e pinze, le seconde di organi di un paziente, quali cuore, fegato, rene e vasi sanguigni addominali. Per la prima tipologia di immagini il risultato della scansione della sonda sono 83 piani, ottenuti facendo coprire alla sonda un arco di 90° e memorizzando un'immagine per ogni grado di rotazione. In pratica, le 83 immagini bidimensionali sono disposte nello spazio come le pagine di un libro aperto con le copertine che formano tra loro un angolo di 90° (Figura 5.3). Le informazioni legate ai pixel che costituiscono l'immagine sono quindi fornite in coordinate polari. La seconda tipologia d'immagini è costituita da oggetti definiti globalmente nello spazio 3D in un sistema di riferimento cartesiano ortonormale; tale immagine è stata ricavata mediante una trasformazione di coordinate (da polari a car-

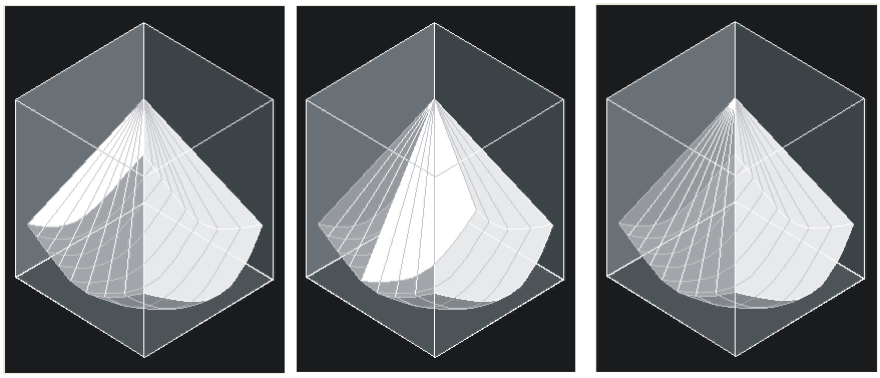

Figura 5.3: In figura è illustrato uno schema del processo eseguito dalla sonda ecografica per la generazione degli 83 piani che costituiscono l'immagine 3D. Partendo da sinistra è evidenziato il primo piano, il centrale e l'ultimo.

tesiane) dell'immagine echografica costituita da piani 2D.
I test eseguiti in questo capitolo sono relativi all'operazione di denoising con le condizioni al contorno di Dirichlet e **BJ** (Block Jacobi) come precondizionatore.

5.3.1 Risultati ottenuti su immagini sintetiche

L'immagine 3D ha dimensione $128 \times 100 \times 83$ ed è stata realizzata immergendo la sonda in una bacinella d'acqua al cui interno sono stati posti due dadi di marmo, uno dei quali ha una gomma pogiata su una faccia (Figura 5.4). Sono stati fissati i parametri $\tau = 1.0e - 03$ (ampiezza del passo di scala) ed $\epsilon = 1.0e - 01$ (parametro di regolarizzazione).
Nella tabella 5.1, per ciascuna scala eseguita, sono indicati il numero di iterazioni lineari (#IL), non lineari (#IN), il numero di valutazioni di funzione (#VF) e la norma del residuio ($\|F(u)\|_2$).

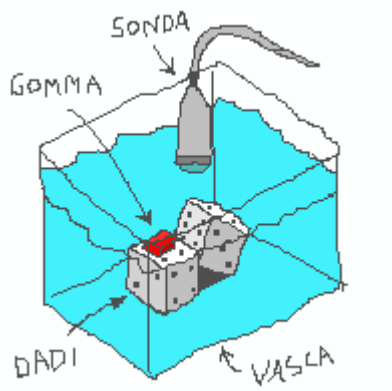

Figura 5.4: In figura è illustrato uno schema degli oggetti presenti nell'immagine test ecografica 3D: due dadi di marno, uno dei quali ha una gomma pogiata su di una faccia.

Nelle Figure 5.5 e 5.6 è indicato il piano 29 dell'immagine 3D dei dadi rispettivamente prima e dopo l'applicazione del processo di denoising. Nella figura 5.7 sono illustrati i grafici dei tempi, espressi in secondi, ottenuti su due distinte architetture: un cluster PC Pentium[2] (linea continua verde) ed un Cluster Hp Itanium[3] (linea continua rossa). Si può notare la differenza importante che c'è nel tempo di esecuzione con un solo processore tra le due architetture: si passa da 1 ora e 30 minuti a 40 minuti. Questo è dovuto principalmente al movimento dei dati dalla memoria principale a quella

[2]Cluster PC Pentium:
Processore : *Pentiunm* 4.1.5, 1.5 *GHz*
Memoria Cache : 256 *Kb*
Memoria Principale : 512 *Mb* (*Rimm*) *PC*800
Network : *Quadricx* – *QsNet* 900 *Mbytes*
[3]Cluster Hp Itanium:
Processore : *ITANIUM BIPROCESSOR* 2.1.4 *GHz*
Memoria Principale : 4 *GB*
Network : *Quadricx* – *QsNet* 900 *Mbytes*

cache, data la dimensione elevata del problema (In ciascun processore la memoria cache è di 256 Kb, mentre quella principale è di 512 Mb). Sul cluster Hp Itanim si può notare come il tempo scali da 40 minuti con un processore a circa 4 minuti con dieci.

Quanto detto per il grafico dei tempi si riflette nel grafico dello speed up (Figura 5.8) in cui sono evidenti sia una situazione di superlinearitá per il cluster PC Pentium, essendo il tempo T_1 (tempo di esecuzione con un solo processore) maggiorato dal non efficiente utilizzo della memoria, sia un andamento lineare per il cluster Hp Itanium. Riguardo lo speedup del cluster Hp Itanium si può notare, inoltre, come pur essendo la macchina costitutita da 64 nodi, ne sono stati utilizzati al più 10 in quanto aumentando ulteriormente p avremmo ottenuto solo un incremento dell'overhead di comunicazione OC_p, a discapito delle prestazioni dell'algoritmo.

	#IL	#IN	#VF	$\|F(u)\|_2$
Scala1	548	11	586	$1.84946e-11$
Scala2	304	7	327	$3.91185e-12$
Scala3	212	5	227	$3.88503e-14$

Tabella 5.1: Nella tabella, fissati $\tau = 1.0e-3$ ed $\epsilon = 1.0e-1$, per ogni passo di scala , sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare.

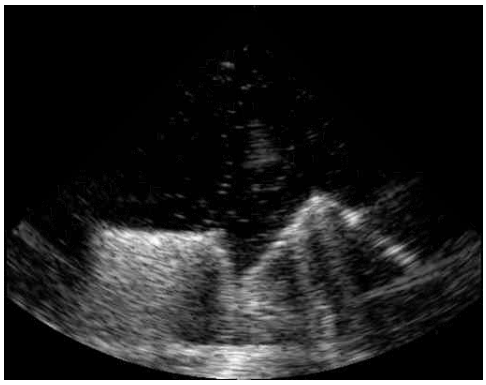

Figura 5.5: Piano 29 dell'immagine ecografica 3D acquisita.

Figura 5.6: Piano 29 dell'immagine echografica 3D dopo il processo di denoising.

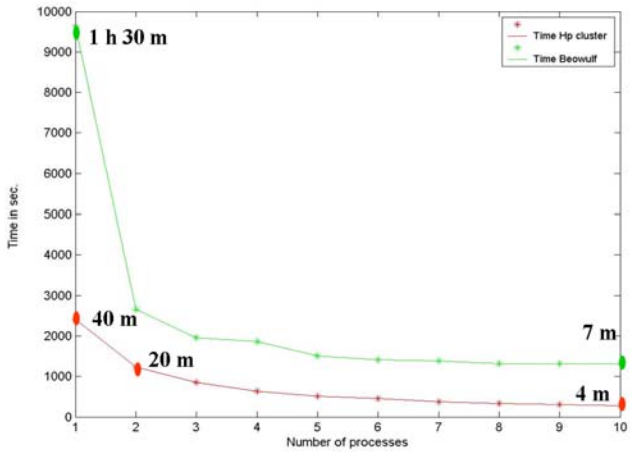

Figura 5.7: In figura sono illustrati i grafici dei tempi, espressi in secondi, relativi ad un'immagine di dimensione $128 \times 100 \times 83$. La linea verde è relativa al cluster PC Pentium, mentre la linea rossa al cluster Hp Itanium.

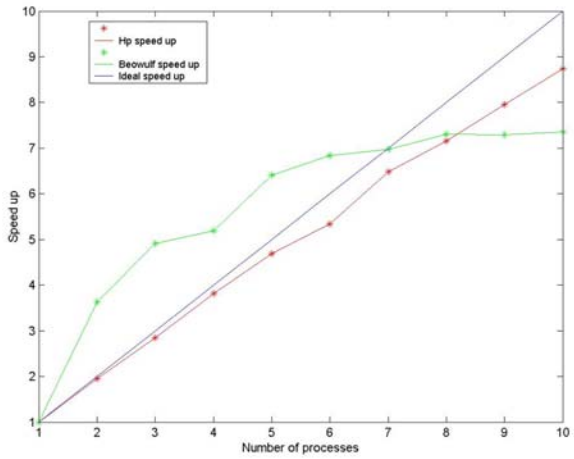

Figura 5.8: In figura sono illustrati i grafici dello speedup relativo ad un'immagine di dimensione $128 \times 100 \times 83$. La linea verde è relativa al cluster PC Pentium, mentre la linea rossa al cluster Hp Itanium.

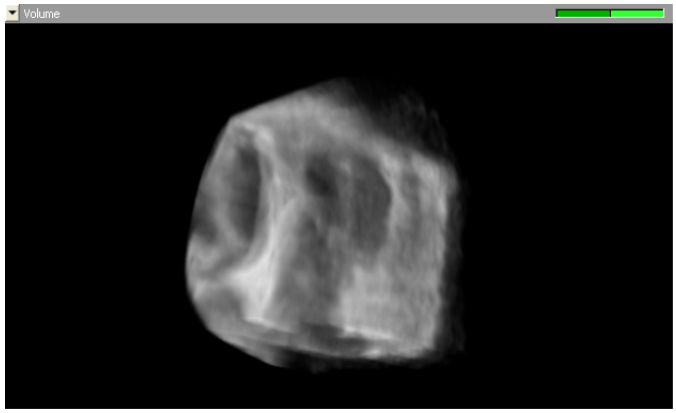

Figura 5.9: In figura è raffigurato il cuore 3D.

5.3.2 Risultati su ecografie reali

L'immagine 3D ha dimensione $151 \times 151 \times 101$ e rappresenta un cuore (Figura 5.9). L'immagine è 3D, è rappresentata in coordinate cartesiane ed è stata ottenuta da una ecografia generata in coordinate polari; le tre sezioni lungo i piani $x-y$ ($z = 50$), $x-z$ ($y = 75$) ed $y-z$ ($x = 75$) sono rappresentate in Figura 5.10.
Nella tabella 5.2 per ciascuna scala temporale eseguita, sono indicati il numero di iterazioni lineari (#IL), non lineari (#IN), il numero di valutazioni di funzioni (#VF) e la norma del residuio ($\|F(u)\|_2$). In fine, nelle Figure 5.11, 5.13 e 5.15 è raffigurato il cuore dopo i tre passi temporali avendo fissato $\tau = 1.0e-3$ ed $\epsilon = 8.0e-1$; nelle Figure 5.12, 5.14 e 5.16 sono rappresentate le corrispondenti sezioni.

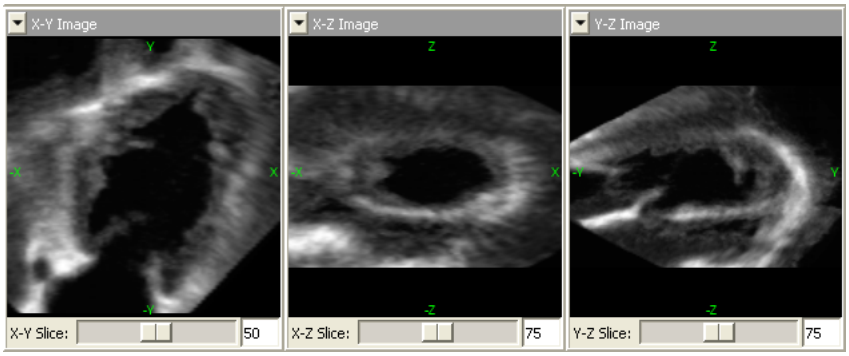

Figura 5.10: In figura sono illustrate le sezioni dei 3 piani: $x-y$ (piano $z=50$), $x-z$ (piano $y=75$) ed $y-z$ (piano $x=75$).

	#IL	#IN	#VF	$\|F(u)\|_2$
Scala1	466	10	500	$6.6537e-11$
Scala2	13769	9	14249	$1.47271e-4$
Scala3	12289	4	12707	$6.9515e-4$

Tabella 5.2: Nella tabella, fissati $\tau = 1.0e-3$ ed $\epsilon = 1.0e-1$, per ciscun passo di scala, sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare.

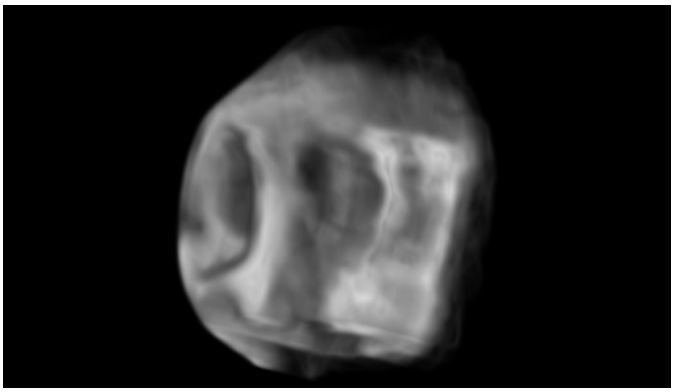

Figura 5.11: In figura è rappresentato il cuore dopo il primo passo di scala.

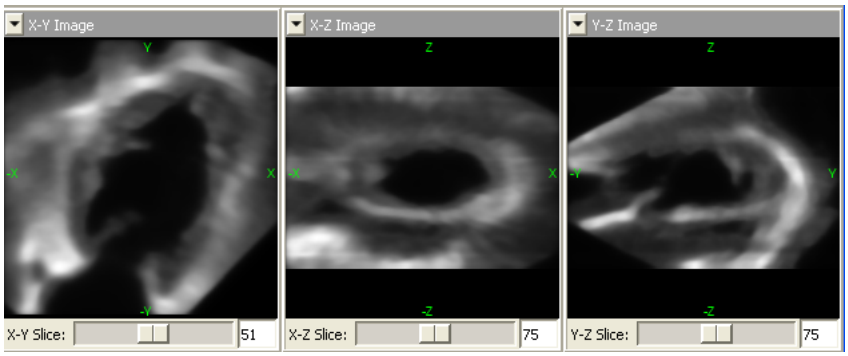

Figura 5.12: In figura sono illustrate le sezioni dei 3 piani dopo il primo passo di scala: $x - y$ (piano $z = 51$), $x - z$ (piano $y = 75$) ed $y - z$ (piano $x = 75$), dopo il processo di denoising.

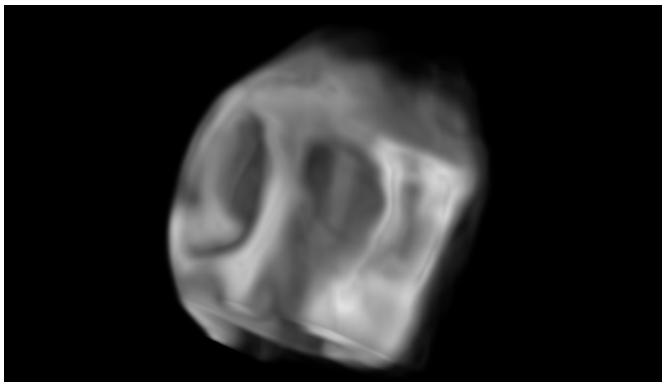

Figura 5.13: In figura è rappresentato il cuore dopo il secondo passo di scala.

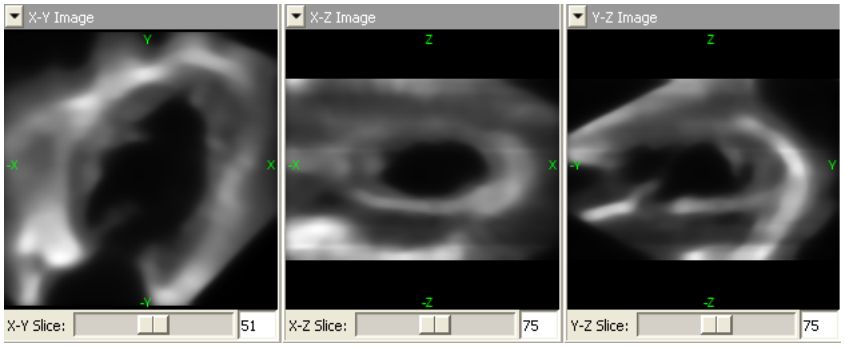

Figura 5.14: In figura sono illustrate le sezioni dei 3 piani dopo il secondo passo di scala: $x-y$ (piano $z = 51$), $x-z$ (piano $y = 75$) ed $y - z$ (piano $x = 75$), dopo il processo di denoising.

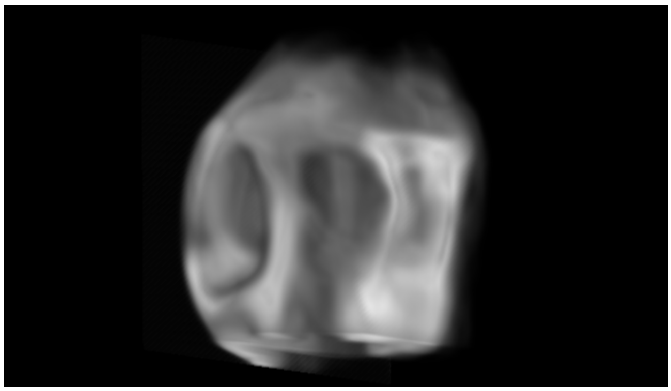

Figura 5.15: In figura è rappresentato il cuore dopo il terzo passo di scala.

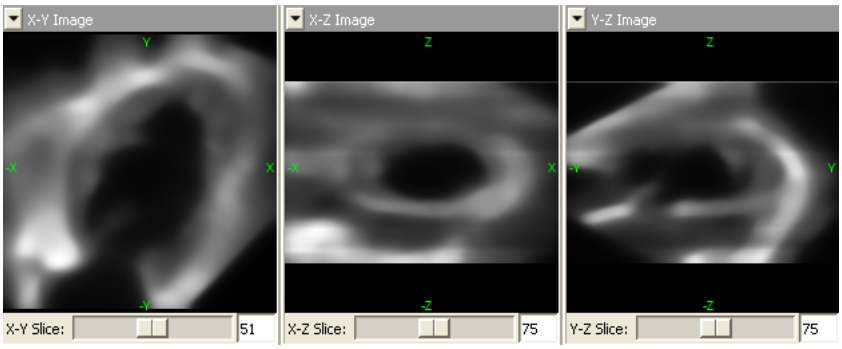

Figura 5.16: In figura sono illustrate le sezioni dei 3 piani dopo il terzo passo di scala: $x - y$ (piano $z = 51$), $x - z$ (piano $y = 75$) ed $y - z$ (piano $x = 75$), dopo il processo di denoising.

Capitolo 6

Esperienze effettuate

6.1 Test eseguiti sul modello non lineare (Schema Implicito)

Nel Gruppo I di Test di questo paragrafo sono descritti i test eseguiti sull'algoritmo sequenziale con **ILU(0)** (fattorizzazione LU Incompleta) come precondizionatore, nella II parte quelli eseguiti in parallelo, **BJ** (Block Jacobi) come precondizionatore.

6.1.1 Gruppo I Test

I test sono stati eseguiti su un'immagine sintetica 3D di dimensione $20 \times 20 \times 20$, costituita da un cilindro schiacciato leggermente nella zona centrale, il cui colore è definito sulla scala di grigio (da 0 a 255) di intensità decrescente procedendo verso le pareti esterne. Nella Figura 6.1 è illustrato l'oggetto 3D, mentre nella Figura 6.2 sono illustrati esempi delle tre sezioni nei piani $x - y$ (piano $z = 9$), $x - z$ (piano $y = 9$) e $y - z$ (piano $x = 9$).

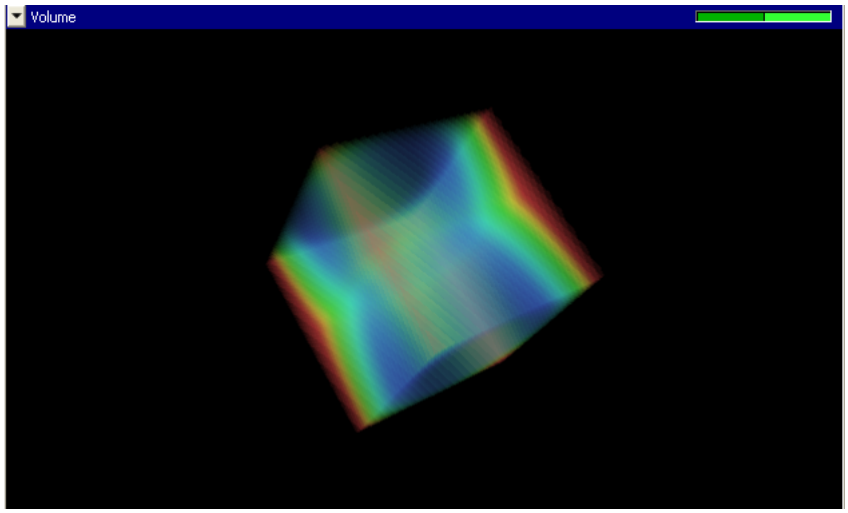

Figura 6.1: La figura rappresenta l'immagine 3D utilizzata per la prima batteria di Test.

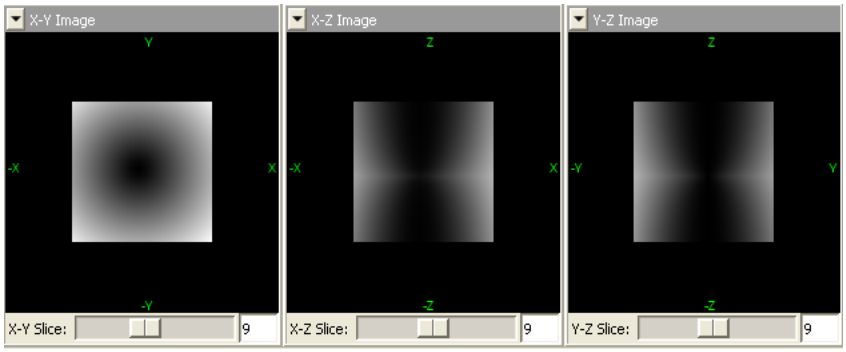

Figura 6.2: La figura rappresenta le tre sezioni dell'immagine 3D utilizzata per la prima batteria di test; piani $x - y$ ($z = 9$), $x - z$ ($y = 9$) e $y - z$ ($x = 9$).

I test eseguiti possiamo suddividerli in due gruppi: il primo in cui l'immagine è priva di rumore e il secondo in cui è stata introdotta una percentuale di rumore additivo (2%, 10% e 20%), con una distribuzione normale con valor medio 0 e varianza 1. Nel primo gruppi di test si sono considerati i due tipi di SNES:

- SNES di tipo FD (Finite Difference) con lo Jacobiano calcolato esplicitamente;
- SNES di tipo MF (Matrix Free) con opportune approssimazioni per lo Jacobiano.

Nel secondo gruppo, visti i risultati ottenuti nei primi test in termini di tempo ed accuratezza, si è posta l'attenzione solo sullo SNES di tipo MF.

Il metodo di Newton si arresta quando $\#IN$ (# **I**terazioni **N**ewton):

- supera il numero massimo di iterazioni (di default $\#IN < 50$);
- oppure supera il numero massimo di **V**alutazioni di **F**unzione (di default $\#VF < 10^4$);
- oppure non si verifica una delle condizioni seguenti:
  - $\|F(u^k)\|_2 < abstol$ (di default $abstol = 1e-50$);
  - $\|u^{k+1} - u^k\|_2 < xtol \cdot \|u^k\|_2$ (di default $xtol = 1e-08$);
  - $\|F(u^k)\|_2 < rtol \cdot \|F(u^0)\|_2$ (di default $rtol = 1e-08$) ;
  - $\|F(u^k)\|_2 = NaN$.

Il criterio di arresto del GMRES si basa su due condizioni:

- $\#IL < 10^4$, dove $\#IL$ indica il numero di **I**terazioni **L**ineari;
- $\|r^k\|_2 < max\,(rtol \cdot \|r^0\|_2, atol)$, con $r^k = b - Ax^k$.

Il metodo verrà considerato divergente se:

$$\|r^k\|_2 > dtol \cdot \|r^0\|_2$$

Di default $rtol = 1e-05$, $atol = 1e-50$, $dtol = 10000$.

Risultati del primo gruppo di test: immagine iniziale priva di rumore

In questo gruppo di test si sono considerate 2 dimensioni per l'immagine 3D[1]: $N = 20$ e $N = 30$. Nella Tabella 6.1 sono indicati i risultati ottenuti per $N = 20$ per entrambi i tipi di SNES e nella Tabella 6.2 quelli relativi a $N = 30$. In entrambi si è posto il passo di scala $\tau = 0.001$.

	#IL	#IN	#VF	Temp	$\|F(u)\|_2$
SNES FD	4	2	16005	$1.012e+03$	$3.89202e-12$
SNES MF	7	2	12	$4.081e+00$	$2.94454e-12$

Tabella 6.1: Nella tabella sono indicati i risultati in dimensione $N = 20$: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; Temp = tempo di esecuzione in secondi; $\|F(u)\|_2$ = norma due della F del sistema non lineare.

	#IL	#IN	#VF	Temp	$\|F(u)\|_2$
SNES FD	4	1	27003	$6.097e+03$	0.000359303
SNES MF	18	3	25	$2.088e+01$	$4.30221e-12$

Tabella 6.2: Nella tabella sono indicati i risultati in dimensione $N = 30$: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; Temp = tempo di esecuzione in secondi; $\|F(u)\|_2$ = norma due della F del sistema non lineare.

Dall'analisi dei tempi di esecuzione risulta evidente come l'utilizzo dello SNES FD è proibitivo già per dimensioni relativamente piccole come $N = 30$. Inoltre si può notare come per $N = 20$ i

[1] La dimensione totale dell'immagine 3D è data da $N \times N \times N$

due metodi siano equivalenti in termini di accuratezza, mentre ciò non si verifica per dimensione $N = 30$ in cui si raggiunge molto velocemente il tetto massimo di valutazioni della funzione dovendo calcolare esplicitamente l'intera matrice Jacobiana. Pertanto nei test successivi prendiamo in esame solo lo SNES MF.

Risultati del primo gruppo di test: Immagine iniziale rumorosa.

Sono state eseguite batterie di test per la determinazione dei valori *"ottimali"* per i parametri τ (ampiezza del passo di scala) ed ϵ (parametro di regolarizzazione) al variare della scala $S = \{1, 2, 3\}$ e della percentuale di rumore introdotta nell'immagine sitetica iniziale (2%, 10% e 20 %).
I risultati più significativi si sono avuti per

$$\begin{cases} \tau \in \{1.0e-3,\ 5.0e-3,\ 1.0e-2\} \\ \epsilon \in \{1.0e-1,\ 1.0e-2,\ 1.0e-3\} \end{cases}$$

La dimensione dell'immagine 3D in tutti i test è $20 \times 20 \times 20$.

Consideriamo i risultati dei test con rumore al 20%.
Nella Tabella 6.3 sono indicati i valori dell'errore relativo (ErrRel) dell'immagine calcolato secondo la formula:

$$ErrRel = \frac{\|ImmIn - ImmFin\|_2}{\|ImmIn\|_2};$$

mentre, nella Tabella 6.4 sono indicati i valori della norma due della funzione F ($||F(x,t)||_2$), al secondo passo di scala e al variare di τ ed ϵ.

Da questi dati si deduce che il valore più basso dell'errore relativo, ed uno tra i più bassi per il residuo, è assunto per $\tau = 5.0e-3$ ed

$\epsilon \setminus \tau$	$1.0e-3$	$5.0e-3$	$1.0e-2$
$1.0e-1$	0.1291	0.0815	0.1095
$1.0e-2$	0.1297	0.0829	0.1092
$1.0e-3$	0.1297	0.0831	0.1093

Tabella 6.3: Nella tabella, fissata la seconda scala, al variare di ϵ e τ viene indicato l'errore relativo dell'immagine (ErrRel).

$\epsilon \setminus \tau$	$1.0e-3$	$5.0e-3$	$1.0e-2$
$1.0e-1$	$5.46387e-11$	$2.50604e-14$	$2.65577e-13$
$1.0e-2$	$8.18600e-11$	$3.51164e-14$	$4.27197e-10$
$1.0e-3$	$1.08658e-12$	$4.90931e-12$	$1.20496e-12$

Tabella 6.4: Nella tabella, fissata la seconda scala, al variare di ϵ e τ viene indicato il residuo del sistema ($||F(x,t)||_2$).

$\epsilon = 1.0e-1$. A conferma di ciò si considerino le Figure 6.3, 6.4 e 6.5 in ciascuna delle quali, fissato il piano $z = 10$, sono illustrati i profili nel piano $x-y$ delle immagini seguenti:

1. non rumorosa (tratto blu);
2. con rumore additivo (tratto verde);
3. ricostruita al primo passo di scala (tratto rosso);
4. ricostruita al secondo passo di scala (tratto nero);
5. ricostruita al terzo passo di scala (tratto giallo).

con il paramtro $\epsilon = 1.0e-1$ e τ che varia nell'insieme seguente:

$$\{1.0e-3,\ 5.0e-3,\ 1.0e-2\}$$

Nelle Figure 6.6, 6.7, 6.8 sono illustrate le corrispondenti curve di livello. Fissato il parametro $\tau = 5 * 1.0e-3$, nelle tabelle 6.5, 6.6,

6.7, al variare di $\epsilon = \{1.0e-1,\ 1.0e-2,\ 1.0e-3\}$, sono indicati, per ciascuno dei tre passi temporali, il numero di iterazioni eseguite per il GMRES (#IL), il numero di iterazioni per il Newton (#IN), il numero di valutazioni di funzione (#VF), la norma due della $F(x,t)$ e l'errore relativo dell'immagine (ErrRel). Nella Tabella 6.8 fissato il parametro $\tau = 5 * 1.0e-3$ e al variare di ϵ viene indicato il tempo di esecuzione complessivo dei tre passi temporali.

Fissato sia $\tau = 5.0e-3$ che $\epsilon = 1.0e-1$, nelle Figure dalla 6.9 alla 6.13 sono illustrate rispettivamente le surfaces di una sezione dell'immagine iniziale priva di rumore, dell'immagine con rumore aggiunto e dell'immagine calcolata ai passi temporali 1, 2 e 3. Inoltre, in Figura 6.14, partendo dall'alto a sinistra e procedendo in senso orario, sono raffigurati i volumi del'immagine rumorosa e di quella ottenuta nei tre passi temporali.

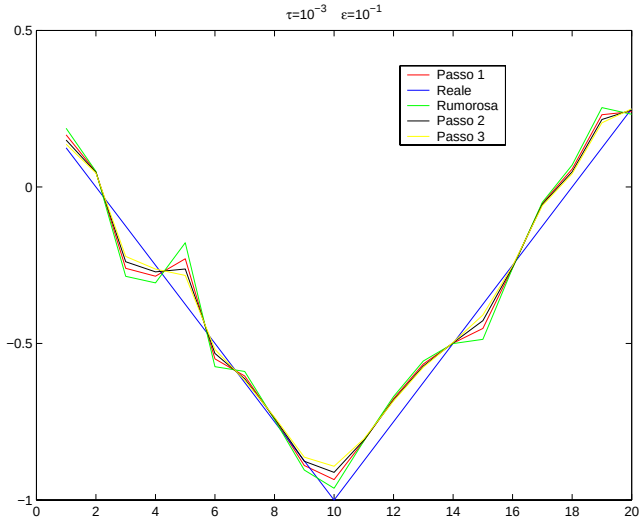

Figura 6.3: In figura sono rappresentati i profili dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 1.0e-3$ ed $\epsilon = 1.0e-3$.

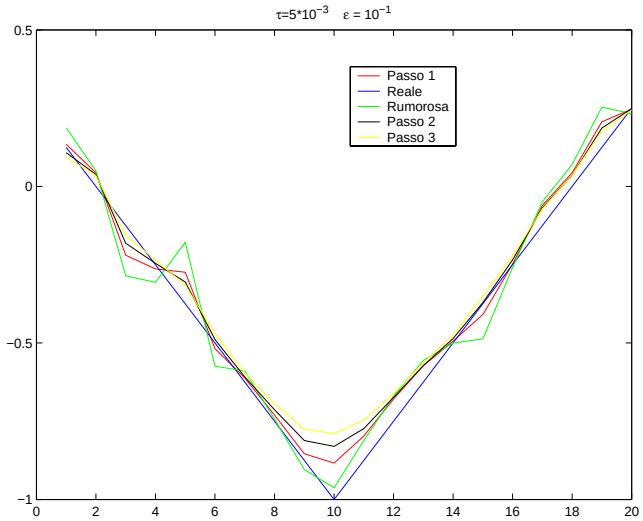

Figura 6.4: In figura sono rappresentati i profili dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 5.0e-3$ ed $\epsilon = 1.0e-3$.

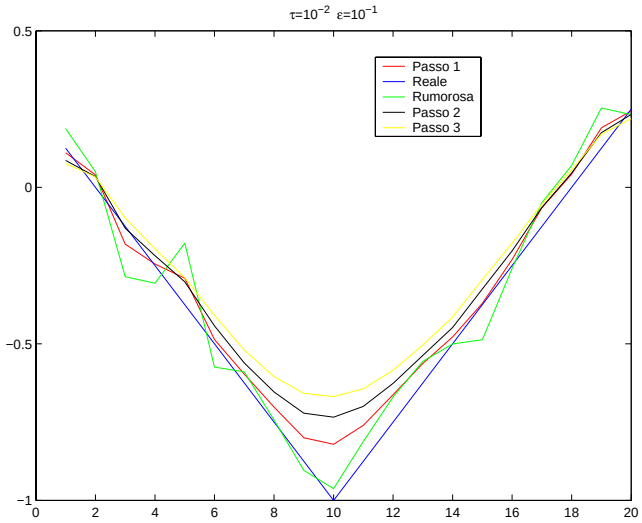

Figura 6.5: In figura sono rappresentati i profili dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 1.0e-2$ ed $\epsilon = 1.0e-3$.

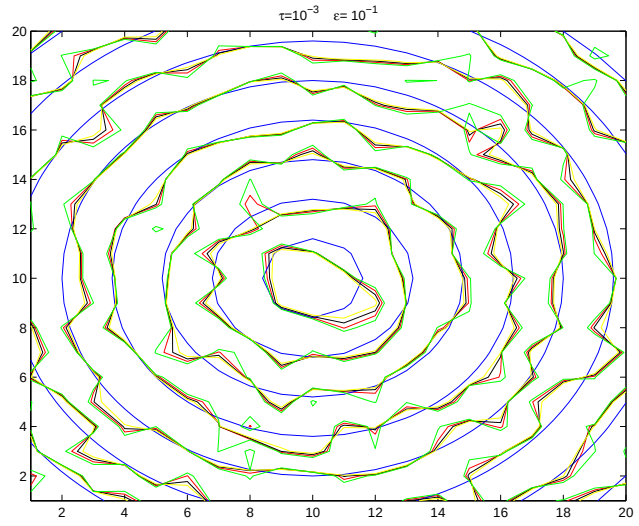

Figura 6.6: In figura sono rappresentate le surfaces dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 1.0e-3$ ed $\epsilon = 1.0e-3$.

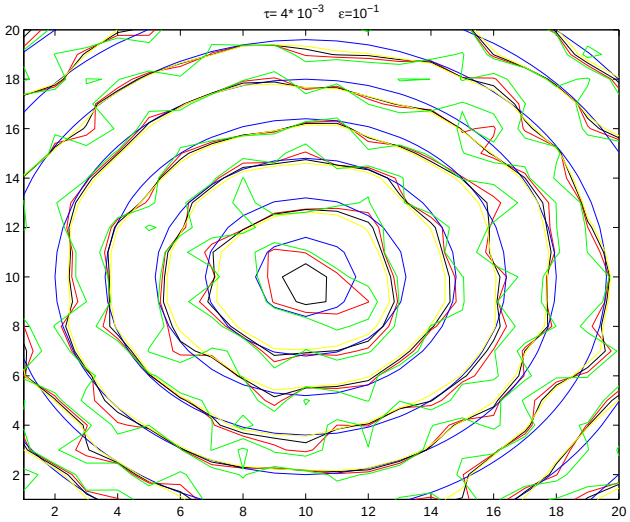

Figura 6.7: In figura sono rappresentate le surfaces dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 5.0e-3$ ed $\epsilon = 1.0e-3$.

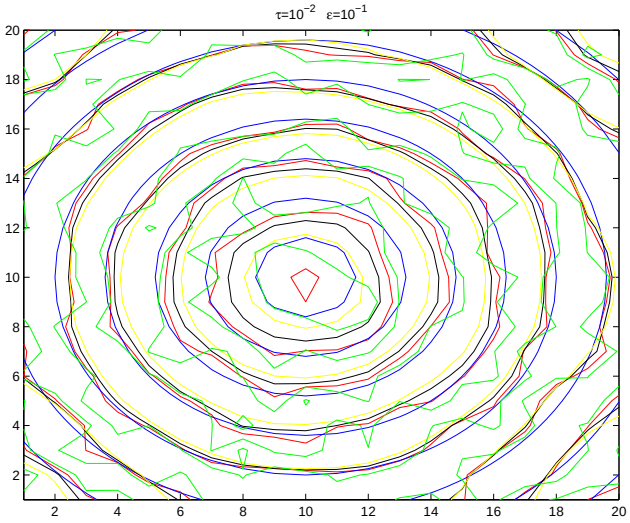

Figura 6.8: In figura sono rappresentate le surfaces dell'immagine sintetica iniziale (tratto blú), rumorosa (tratto verde), alaborata al passo 1 (tratto rosso), al passo 2 (tratto nero) ed al passo 3 (tratto giallo). Sono stati fissati $\tau = 1.0e-2$ ed $\epsilon = 1.0e-3$.

	#IL	#IN	#VF	$\|F(u)\|_2$	ErrRel
Scala1	18	4	27	$1.1705e-10$	0.1090
Scala2	20	4	29	$2.50604e-14$	0.0815
Scala3	15	3	22	$6.41203e-11$	0.0832

Tabella 6.5: Nella tabella, fissati $\tau = 5 * 1.0e-3$ ed $\epsilon = 1.0e-1$, per ogni passo di scala, sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare; ErrRel = errore relativo dell'immagine.

	#IL	#IN	#VF	$\|F(u)\|_2$	ErrRel
Scala1	20	5	31	$4.67444e-11$	0.1101
Scala2	16	4	25	$3.50436e-10$	0.0829
Scala3	16	4	25	$4.96844e-14$	0.0840

Tabella 6.6: Nella tabella, fissati $\tau = 5 * 1.0e-3$ ed $\epsilon = 1.0e-2$, per ogni passo di scala sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare; ErrRel = errore relativo dell'immagine.

	#IL	#IN	#VF	$\|F(u)\|_2$	ErrRel
Scala1	24	6	37	$8.17085e-13$	0.1102
Scala2	20	5	31	$2.68496e-13$	0.0831
Scala3	16	4	25	$2.04699e-11$	0.0841

Tabella 6.7: Nella tabella, fissati $\tau = 5 * 1.0e-3$ ed $\epsilon = 1.0e-3$, peer ogni passo di scalal sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare; ErrRel = errore relativo dell'immagine.

τ	ϵ	Tempo in secondi
$5 * 1.0e-3$	$1.0e-1$	$9.1350e+00$
$5 * 1.0e-3$	$1.0e-2$	$1.0463e+01$
$5 * 1.0e-3$	$1.0e-3$	$1.2275e+01$

Tabella 6.8: Nella tabella, fissato $\tau = 5 * 1.0e-3$, vengono indicati il parametro ϵ ed il tempo di esecuzione complessivo delle tre scale temporali.

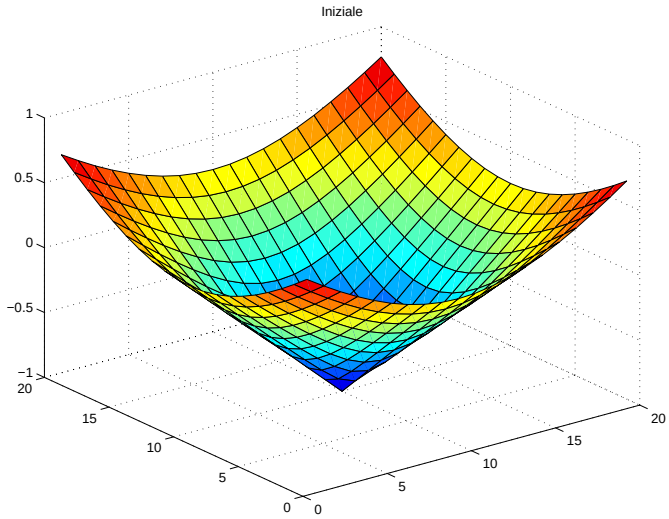

Figura 6.9: Surface del piano 29 dell'immagine iniziale.

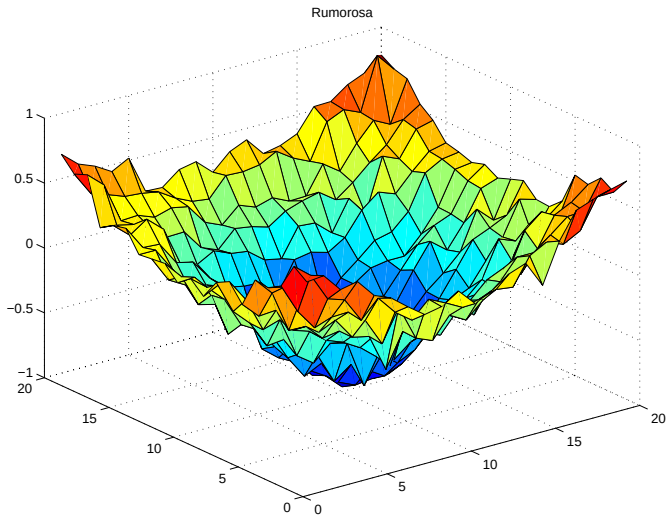

Figura 6.10: Surface del piano 29 dell'immagine rumorosa.

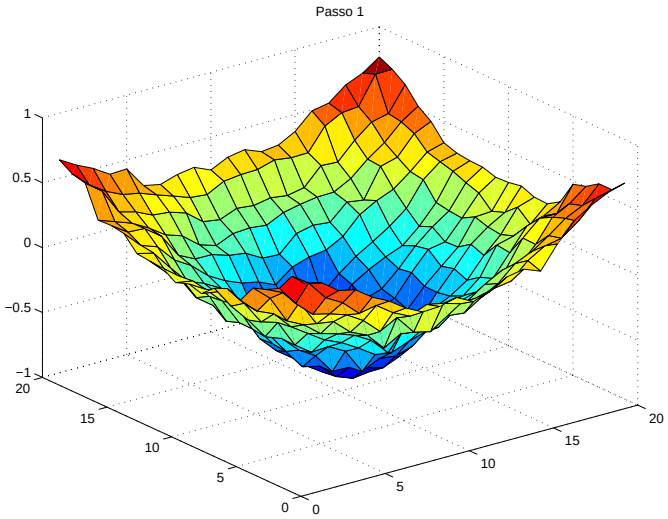

Figura 6.11: Surface del piano 29 dell'immagine dopo il primo passo di scala.

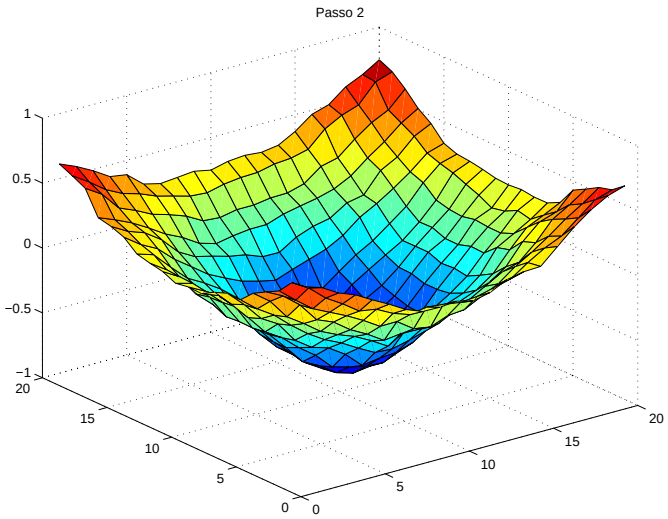

Figura 6.12: Surface del piano 29 dell'immagine dopo il secondo passo di scala.

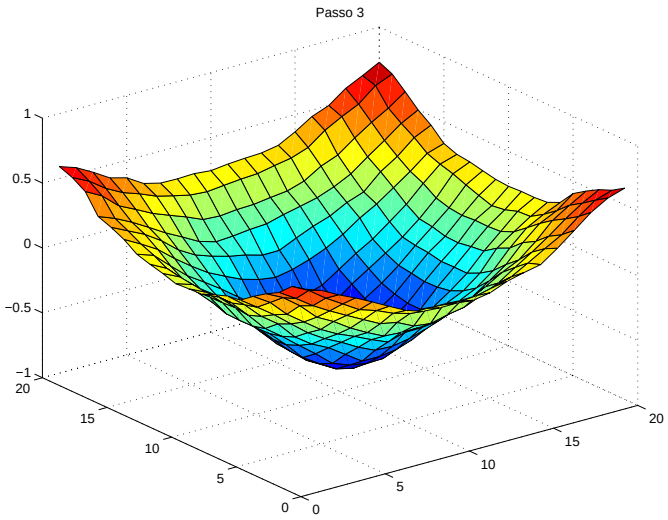

Figura 6.13: Surface del piano 29 dell'immagine dopo il terzo passo di scala.

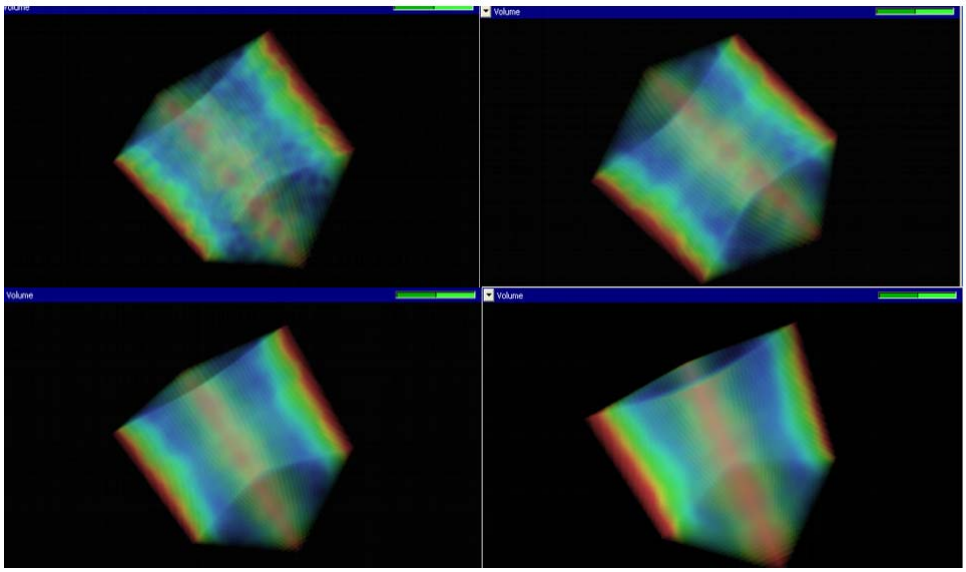

Figura 6.14:

6.1.2 Gruppo II di Test

I test sono stati eseguiti su un'immagine sintetica 3D di dimensione $70 \times 70 \times 70$, costituita da ellissoidi con assi coincidenti il cui colore è definito sulla scala di grigio (da 0 a 255) di intensità decrescente procedendo verso l'esterno. Nella Figura 6.15 è illustrato l'oggetto 3D, mentre nella Figura 6.16 sono illustrati esempi delle tre sezioni nei piani $x - y$ (piano $z = 35$), $x - z$ (piano $y = 34$) e $y - z$ (piano $x = 34$).
Fissato il parametro $\tau = 1.0e - 3$ ed $\epsilon = 5 * 1.0e - 2$ nella tabella 6.9 sono indicati, per ciascuno dei tre passi temporali, il numero di iterazioni eseguite per il GMRES (#IL), il numero di iterazioni per il Newton (#IN), il numero di valutazioni di funzione (#VF) e la norma due della $F(x, t)$.

Fissato sia $\tau = 5.0e - 3$ che $\epsilon = 1.0e - 1$, nella Figura 6.17 è rappresentato il piano $z = 35$ dell'immagine 3D con le relative curve di livello. Partendo in alto a sinistra e procedendo in senso orario si hanno le curve di livello dell'immagine iniziale, di quella rumorosa e di quella calcolata ai passi temporali 1 e 2.
In fine, in Figura 6.18, sempre partendo dall'alto a sinistra e procedendo in senso orario, sono raffigurati i volumi dell'immagine rumorosa e di quella ottenuta nei tre passi temporali.

	#IL	#IN	#VF	$\|F(u)\|_2$
Scala1	842	9	886	$3.24042e - 13$
Scala2	143	5	156	$4.07599e - 12$
Scala3	173	5	186	$7.7532e - 12$

Tabella 6.9: Nella tabella, fissati $\tau = 5 * 1.0e - 3$ ed $\epsilon = 1.0e - 3$, per ogni passo di scala sono indicati: $\#IL$ = numero di iterazioni lineari; $\#IN$ = numero di iterazioni di Newton; $\#VF$ = numero di valutazioni di funzione; $\|F(u)\|_2$ = norma due della F del sistema non lineare.

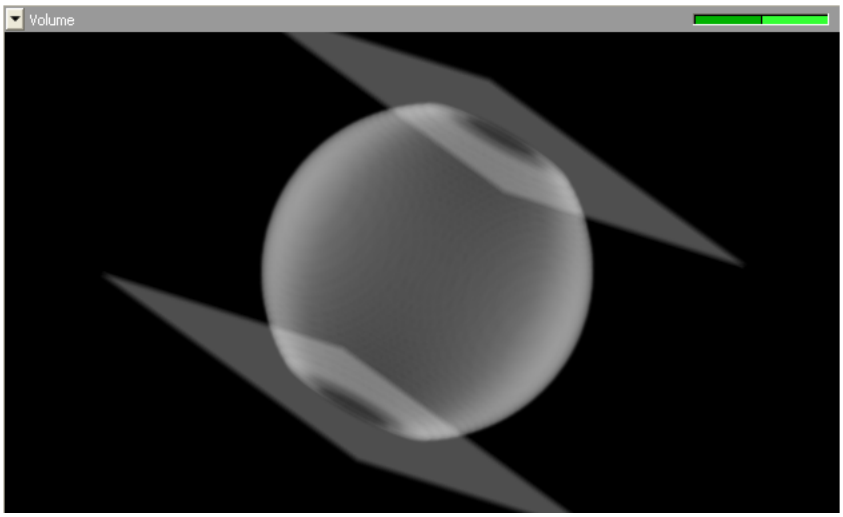

Figura 6.15: La figura rappresenta l'immagine 3D utilizzata per la seconda batteria di Test.

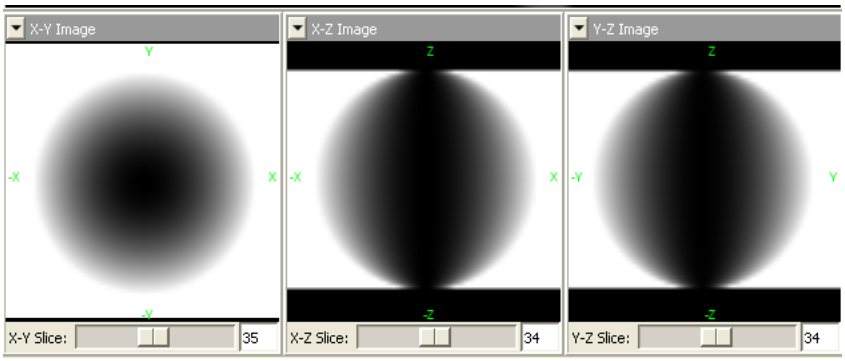

Figura 6.16: La figura rappresenta le tre sezioni dell'immagine 3D utilizzata per la seconda batteria di test; i piani $x-y$ ($z=35$), $x-z$ ($y=34$) e $y-z$ ($x=34$).

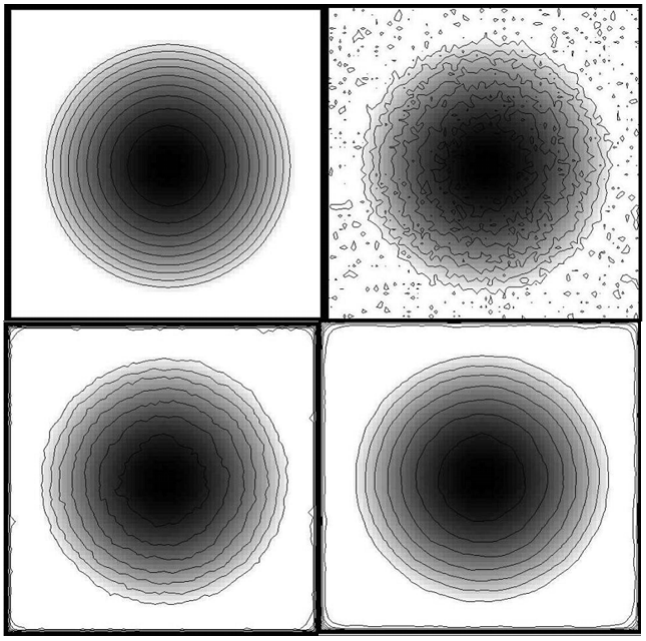

Figura 6.17: La figura rappresenta il piano $z = 35$ dell'immagine 3D con le relative curve di livello. Partendo in alto a sinistra e procedendo in senso orario si hanno le curve di livello dell'immagine iniziale, di quella rumorosa e di quella calcolata ai passi temporali 1 e 2.

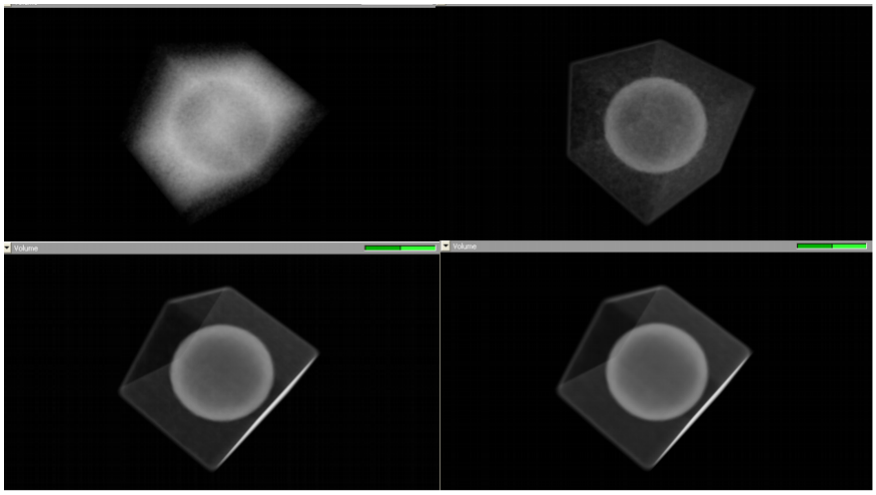

Figura 6.18: Partendo in alto a sinistra e procedendo in senso orario la figura rappresenta l'immagine 3D rumorosa e quella calcolata ai passi temporali 1, 2 e 3.

Bibliografia

[1] L. Alvarez, F. Guichard, P. L. Lions, J. M. Morel - *Axioms and Fundamental Equations of Image Processing* - Archive for Rat. Mech. Anal., Vol. 123, 1993, pp. 200-257.

[2] L. Alvarez, P. L. Lions, J. M. Morel - *Image selective smoothing and edge detection by nonlinear diffusion II* - SIMAI J. Numer. Anal., Vol. 29, 1992, pp. 182-193.

[3] L. Alvarez, J. M. Morel - *Formalization and computational aspects of image analysis* - Acta Numerica, 1994, pp. 1-59.

[4] G. Aubert, P. Kornprobst - *Mathematical problems in image processing* - Applied Mathematical Sciences, vol. 147, 1996.

[5] M. Benzi - *Preconditioning Techniques for Large Linear Systems: A Survey* - Journal of Computational Physics 182, 418-477 (2002).

[6] P.N. Brown - *A local convergence theory for combined inexact-Newton/finite -difference projection methods.* - SIAM J. Numer. Anal. , 24, 1987, pp. 407-434.

[7] P. N. Brown and A. C. Hindmarsh - *Matrix-free methods for stiff systems of ode's* - SIAM J. Sci. Stat. Comput., Vol 23, No. 3, pp 610-638, June 1986.

[8] P. N. Brown and Yousef Saad - *Hybrid Krylov methods for nonolinear systems of equations* - SIAM J. Sci. Stat. Comput., Vol 11, No. 3, pp 450-481, May 1990.

[9] P. N. Brown and Yousef Saad - *Convergence theory for nonlinear Newton-Krylov algoritms* - SIAM J. Opt., vol. 4, 1994, pp. 297-330.

[10] V. Caselles, R. Kimmel, G. Sapiro - *Geodesic Active Countours* - International Journal of Computer Vision, Vol. 22, pp. 61-79, 1997.

[11] V. Caselles, R. Kimmel, G. Sapiro, C. Sbert - *Minimal Surfaces: A Three Dimensional Segmentation Approch* - Numer. Math., Vol 77, pp. 423-451, 1997.

[12] F. Catté, P.L. Lions, J.M. Morel T. Coll- *Image selective smoothing and edge detection by nonlinear diffusion* - SIMAI J. Numer. Anal., Vol. 29 No. 1, pp. 182-193, February 1992.

[13] T. F. Chan and K. R. Jackson - *Nonlinear preconditioned Krylov subspace methods for discrete Newton algorithms* - SIAM J. Sci. Stat. Comput., Vol 5, No. 3, pp 533-542, September 1984.

[14] Y.-G. Chen, Y. Giga and S. Goto - *Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations* - Proc. Japan Acad. Ser. A. Math. Sci.,Vol 65, pp. 207-210, 1989.

[15] Y.-G. Chen, Y. Giga and S. Goto - *Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations* - J. Differential Geometry, Vol 33, pp. 749-786, 1991.

[16] S. Corsaro, K. Mikula, A. Sarti, F. Sgallari - *Semi-implicit co-volume method in 3D image segmetation* - Preprint, Department of Mathematics, Slovak University of Technology, Bratislava (2004).

[17] Ron S. Dembo, S. C. Eisenstat, T. Steihaug - *Inexact newton Methods* - SIAM J. Sci. Stat. Comput., Vol 19, No. 2, pp 400-408, April 1982.

[18] J.E. Dennis, R.B. Schnabel - *Numerical Methods for Uncostrained Optimization and Numerical Equations.* - Prentice-Hall, Englewood Cliffs, NJ, 1983.

[19] S.C. Eisenstat, H.F. Walker - *Globaly convergent inexact Newton methods* - SIAM J. Optim., vol. 4, 1994, pp. 393-422.

[20] L.C. Evans and J. Spurck - *Motion of level sets by mean curvature. I* - J. Differential Geom., Vol 33, 635-681, 1991.

[21] R. Eymard, T. Gallouet, R. Herbin - *Finite volume Methods* - in: Handbook for Numerical Analysis 7 (Ph. Ciarlet, P.L. Lions, eds), Elsevier, 2000.

[22] P. Frolkovic, K. Mikula - *Flux-based level set methods: a finite volume method for evolving interfaces* - Preprint IWR/SFB No. 2003-15 , Interdisciplinary Center for Scientific Computing, University of Heidelberg (2003).

[23] C.W. Gear and Y. Saad - *Iterative solution of linear equations in ode codes* - SIAM J. Sci. Stat. Comput., Vol 4, No. 4, pp 583-601, December 1983.

[24] F. Guichard e J.-M. Morel -*Image iterative smooting and PDE's* - Online Book, September 2000;

[25] A. Handlovicova, K. Mikula, A. Sarti - *Numerical solution of parabolic equations related to level set formulation of mean curvature flow* - Comput. Visual. Sci. 1:179-182 (1998).

[26] A. Handlovicova, K. Mikula, F. Sgallari - *Semi-implicit complementary volume scheme for solving level set like equations*

in image processing and curve evolution - Numer. Math. (2003)93:675-695, Digital Obect Identifier (DOI).

[27] M. R. Hestenes, E.L. Stiefel - *Methods of conjugate gradients for solving linear sistem* - J. Res. Natl. Bureau of Standards, Section B 49, 1952, pp. 409-436.

[28] C.T. Kelly - *Iterative Methods for Linear and NonLinear Equations* - 1995, Philadelphia.

[29] S. Kichenassamy, A. Kumura, P. Olver, A. Tannenbaum, A. Yezzi - *Conformal curvature flows: from phase transitions to active vision* - Arch. Rat. Mech. Anal., Vol. 134, pp. 275-301, 1996.

[30] D.A. Knoll, D. E. Keyes - *Jacobian-free Newton-Krylov methods: a survey of approaches and applications* - Journal of Computational Physics 193 (2004) 357-397.

[31] J. J. Koenderink - *The structure of images* - Biol. Cibern. 50, 1984, pp. 363-370.

[32] K. Mikula - *Image processing with partial differential equations* - in Modern Methods in Scientific Computing and Applications (A.Bourlioux and M.Gander,Eds.), NATO Science Ser. II, Vol. 75, Kluwer Academic Publishers, Dodrecht, 2002, pp. 283-322.

[33] K. Mikula - *Computational solution, applications and analysis of soma geometrical nonlinear diffusion equations* - Isaac Newton Institute for Mathematical Sciences, University of Cambridge, Preprint No. NI03027-CPD (2003).

[34] K. Mikula, N. Ramarosy - *Semi-implicit finite volume scheme for solving nonlinear diffusion equations in imaging processing* - Numer. Math. (2001) Digital Obect Identifier (DOI).

[35] K. Mikula, A. Sarti, C. Lamberti - *Geometrical Diffusion in 3D-Echocardiography* - in Procedings of ALGORITMY'97, Conference on Scientific Computing, September 1997 (pp. 167-181).

[36] K. Mikula, A. Sarti, F. Sgallari - *Co-volume level set method in subjective surface based medical image segmentation* - in: Handbook of Medical Image Analysis: Segmentation and Registration Models (J.Suri et al., Eds.), Kluwer Academic/Plenum Publishers, New York, 2004.

[37] K. Mikula, A. Sarti, F. Sgallari - *Co-Volume method for Riemannian mean curvature flow in subjective surfaces multiscale segmentation* - Preprint, Department of Mathematics and Descriptive Geometry, Slovak University of Technology (2002).

[38] K. Mikula, F. Sgallari - *Semi-implicit finite volume scheme for image processing in 3D cylindrical geometri* - Numerische Mathematik 89, No.3 (2001), pp. 561-590.

[39] S. Osher, J. Sethian - *Fronts propagating with curvature dependent speed: algorithm based on Hamilton-Jacobi formulation.* - J. Comput. Phys. , Vol. 79, 1988, pp. 12-49.

[40] P. Perona, J. Malik - *Scale space and edge detection using anisotropic diffusion* - Proc. IEEE Computer Society Workshop on Computer Vision (1987).

[41] E. S. Posmentier - *The generation of salinity finestructure by vertical diffusion* - J. Phys. Oceanogr., Vol. 7, pp. 298-300, 1977.

[42] PETSc User Manual - Mathematics and Computer Science Division *http://www.mcs.anl.gov/petsc.*

[43] J.K. Reid - *On the method of conjugate gradients for the solution of large sparse systems of linear equations.* - in: J.K. Reid

(Ed.), Large Sparse Sets of Linear Equations, Academic Press, New York, 1971, pp 231-254.

[44] Yousef Saad - *Iterativ Methods for Sparse Linear Systems* - PWS Publishing Company, Boston, 1996.

[45] Y. Saad, M.H. Schultz - *GMRES: a generalized minimal residual algorithm for solving non symetric linear sistems* - SIAM J. Sci. Stat. Comput., Vol. 7, 1986.

[46] A. Sarti and G. Citti - *Subjective Surfaces and riemannian mean curvature flow of graphs* - Acta. Math. Univ. Comenianae, Vol. LXX, 1(2001), pp. 85-103, Procedings of Algoritmy 2000.

[47] A. Sarti, K. Mikula, F. Sgallari - *Nonlinear Multiscale Analysis of 3D Echocardiographic Sequences* - IEEE Transactions on Medical Imaging, Vol. 18, No.6, 1999, pp. 453-467.

[48] J. Sethian - *Numerical alghoritm for propagating interfaces: Hamilton-Jacobi equations and conservation laws* - J. Differential. Geom., Vol. 31, 1990, pp. 131-161.

[49] J. Weickert - *Anisotropic Diffusion in Image Processing* - ECMI Series, Teubner, Stuttgart, 1998.

[50] A. P. Witkin - *Scala-space filtering* - in: Proc. Eight Internat. Conf. on Artificial Inteligence, Vol. 2, 1983, pp. 1019-1022.