SAPIENZA
UNIVERSITÀ DI ROMA

Master II Livello in "Calcolo Scientifico"

A.A. 2012-2013

Tecniche Variazionali per la Decomposizione di Immagini Astronomiche

Rapporto Finale di Stage
di

Daniele Ottaviani

Tutor interno
Prof. Maurizio Falcone

Tutor esterno (INAF)
Dr. Marco Castellano

Indice

Introduzione **3**

1 Decomposizione di immagini mediante variazione totale **5**
1.1 La variazione totale di una funzione 5
1.2 Rimozione del rumore tramite variazione totale 7
1.3 Decomposizione di immagini astronomiche 9

2 Algoritmi di decomposizione e di selezione parametri **13**
2.1 I tre problemi di minimizzazione 13
2.2 Discretizzazione 15
2.3 L'algoritmo di proiezione di Chambolle 16
2.4 L'algoritmo di Aujol-Aubert-Blanc-Féraud-Chambolle (A^2BC) 17
2.5 L'algoritmo di Soft Thresholding 19
2.6 L'algoritmo di correlazione: selezione automatica del parametro di splitting 21
2.6.1 La relazione tra μ e λ 24
2.7 Il Programma 28
2.7.1 Compilazione e Makefile 28
2.7.2 Esecuzione di ATVD 30
2.7.3 Il file `.REPORT` 35
2.7.4 Un esempio di applicazione 36
2.7.5 Analisi della scalabilità 45

3 Simulazioni numeriche **51**
3.1 Lena: Immagine fotografica con aggiunta di rumore gaussiano 51
3.2 Immagini geometriche: effetto degli algoritmi sui valori dei pixel 58

3.3 Immagini astronomiche simulate 67
3.3.1 Risoluzione con metodo $TV - L^2$ 74
3.3.2 Risoluzione con metodo $TV - L^1$ 76
3.3.3 Risoluzione con metodo $TV - G$ 78
3.3.4 Un confronto tra i risultati 80
3.3.5 Selezione di sorgenti sull'immagine struttura 82
3.4 Decomposizione di immagini astronomiche reali 84
3.4.1 Individuazione di nuove sorgenti nell'immagine ultra-fonda GS-HUDF 93

Conclusioni **96**

Introduzione

Riuscire a decomporre un'immagine in due parti, una contenente le informazioni relative alla struttura dell'immagine stessa (la "parte struttura"), l'altra contenente i dettagli meno significativi, detta "parte texture", è uno dei problemi più importanti e impegnativi del trattamento delle immagini. Dare una corretta definizione di questo tipo di decomposizione è molto difficile. Il concetto che sta alla base è che ogni immagine può essere sempre vista come somma di due parti distinte, una contenente gli oggetti più grandi e importanti, l'altra contenente i dettagli più piccoli, comunemente contraddistinti da comportamenti di natura periodica o oscillatoria. La definizione di "texture" è molto vaga e dipende molto anche dalla scala dell'immagine. Lo stesso soggetto può risultare un componente della parte struttura di un'immagine con una determinata gradazione, mentre può risultare componente della parte texture se si analizza la stessa immagine con una scala differente.

In letteratura esistono due diversi tipi di approccio per risovere questo tipo di problemi, entrambi equivalentemente efficaci e soddisfacenti. Il primo è basato sulla manipolazione dei coefficienti *wavelet* dell'immagine (per approfondimenti consultate [11, 15, 8, 14, 13, 16]). Il secondo invece si propone di risolvere il problema cercando la soluzione di particolari equazioni differenziali alle derivate parziali associate alla minimizzazione di un determinato funzionale energia composto da una qualche norma del gradiente ([20, 7, 3, 16, 18, 19]). Quando la decomposizione viene applicata per rimuovere elementi di disturbo (*rumore*) dall'immagine si parla di problemi di *denoising*.

Lo scopo di questo lavoro è quello di sperimentare algoritmi che riescano a risolvere problemi di denoising applicati alle immagini astronomiche. L'approccio che sceglieremo tra quelli proposti nel paragrafo precedente è

il secondo; in particolare risolveremo il problema sfruttando metodi appartententi alla famiglia di algoritmi che operano tramite tecniche di variazione totale, di cui parleremo più avanti. Trovare un algoritmo soddisfacente che permetta di ottenere un'immagine astronomica il più possibile priva di elementi di disturbo sarebbe di enorme utilità per l'analisi scientifica delle osservazioni astronomiche. Lo sfruttamente scientifico delle immagini ottenute dai telescopi è infatti essenzialmente limitato da disturbi in esse presenti, e in particolare dai livelli di rumore.

Nel Cap.1 di questa tesi presenteremo una breve ma esauriente descrizione delle tecniche di decomposizione tramite variazione totale applicate ad immagini "generiche" e un'introduzione alle immagini astronomiche. Successivamente, nel Cap.2, descriveremo il problema di minimizzazione della variazione totale nel continuo, studiandone tre varianti. Dopo essere passati da spazi continui a spazi discreti, proporremo diversi metodi di soluzione e li implementeremo in un unico programma C++ (Cap.2). Nello stesso capitolo verrà anche presentato un metodo di selezione automatica per il parametro fondamentale nella decomposizione di immagini, il parametro di splitting λ, oltre ad un'accurata descrizione del programma utilizzato, il cui scopo sarà sia quello di aiutare a comprenderne meglio il funzionamento sia quello di mostrare un'analisi della sua efficienza. Infine, nell'ultimo capitolo (Cap.3) presenteremo i risultati del codice su simulazioni numeriche e immagini astronomiche sia simulate che reali.

In particolare osserveremo come l'applicazione di tale decomposizione possa realmente aiutare nella ricerca di oggetti celesti precedentemente non identificabili a causa del rumore di fondo. Nelle ultime due sezioni del Cap.3 affronteremo infatti il problema di decomposizione applicato a immagini astronomiche ottenute attraverso la reale osservazione di una porzione di cielo mediante il telescopio `HUBBLE`. Vedremo come grazie alla decomposizione sarà possibile identificare nuovi corpi celesti.

Capitolo 1

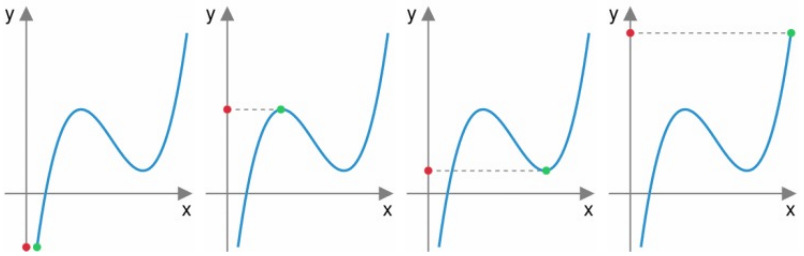
Decomposizione di immagini mediante variazione totale

In questo primo capitolo introdurremo i concetti di base della decomposizione tramite variazione totale. Parleremo brevemente del concetto di variazione totale, del principio che rende efficaci i metodi di decomposizione, del tipo di problemi che intendiamo risolvere e dell'importanza di alcune loro caratteristiche. In ultimo, cercheremo di descrivere brevemente le proprietà delle immagini astronomiche. Per una descrizione più accurata dei problemi e degli algoritmi risolutivi si faccia riferimento il capitolo successivo.

1.1 La variazione totale di una funzione

Sia f una funzione continua e limitata, definita in un intervallo $[a, b] \in \mathbb{R}$ a valori in $\mathbb{R}$. Immaginiamo un suo possibile grafico e pensiamo ad una pallina che, partendo dal punto $(a, f(a))$, percorre tutta la curva (rappresentata dal grafico di f) fino a portarsi nel punto $(b, f(b))$ (vedi Fig. 1.1). Immaginiamo ora di proiettare punto per punto tutto il percorso effettuato dalla pallina sull'asse delle ordinate. La *variazione totale* di f nell'intervallo $[a, b]$ è data dalla lunghezza totale del percorso così ottenuto.

La variazione totale di una funzione continua e limitata definita su un sottoinsieme chiuso di $\mathbb{R}^n$, $n \geq 1$, è formalmente descritta da queste due definizioni;



(a) A: Inizio (b) B (c) C (d) D: Fine

Figura 1.1: La variazione totale della funzione in Fig. è data dalla lunghezza totale del percorso effettuato dalla pallina rossa, la cui posizione punto per punto si ottiene proiettando la pallina verde sull'asse delle ordinate.

Definizione 1.1.1 (**Variazione totale di una funzione con un'unica variabile**)**.** La variazione totale di una funzione continua e limitata f, definita su un intervallo $[a, b] \subset \mathbb{R}$ è data dalla seguente quantità:

$$V_b^a(f) = \sup_P \sum_{i=0}^{n_P-1} |f(x_{i+1}) - f(x_i)|, \tag{1.1}$$

dove la ricerca dell'estremo superiore è da svolgersi nell'insieme

$$\mathcal{P} = \{P = \{x_0, \ldots, x_{n_P}\} | P \text{ è una partizione di } [a, b]\},$$

ovvero l'insieme di tutte le possibili partizioni di $[a, b]$.

Definizione 1.1.2 (**Variazione totale di una funzione a più variabili**)**.** Sia Ω un aperto di $\mathbb{R}^n$. Sia $f \in L^1(\Omega)$. La variazione totale di f in Ω è data dalla seguente quantità:

$$V(f, \Omega) := \sup \left\{ \int_\Omega f(x) \mathrm{div}\phi(x)\, \mathrm{d}x \colon \phi \in C_c^1(\Omega, \mathbb{R}^n),\ \|\phi\|_{L^\infty(\Omega)} \leq 1 \right\}, \tag{1.2}$$

dove $C_c^1(\Omega, \mathbb{R}^n)$ è l'insieme delle funzioni vettoriali continue e differenziabili a supporto compatto contenuto in Ω e $\| \cdot \|_{L^\infty(\Omega)}$ è la norma dell'estremo superiore.

Se si pensa ad f come ad un'immagine (o più in generale ad un segnale), è logico pensare che, essendo la variazione totale un concetto molto legato alla

continuità dei valori contenuti nei pixel, segnali contenenti informazioni abbastanza omogenee tra loro (come un'immagine senza rumore rappresentante qualunque cosa più grande di un pixel) avranno bassi valori di variazione totale. L'idea che sta alla base degli algoritmi di denoising che tratteremo è proprio questa: il rumore aumenta la variazione totale.

Le definizioni appena date suggeriscono che in qualche modo il concetto di variazione totale debba essere legato al concetto di gradiente. Esistono due teoremi che legano questi concetti.

Teorema 1.1.3. *Sia f continua con derivata prima Riemann-integrabile in $[a, b] \in \mathbb{R}$. Allora la sua variazione totale può essere espressa anche dalla seguente formula:*

$$V_b^a(f) = \int_a^b |f'(x)| \mathrm{d}x. \tag{1.3}$$

Teorema 1.1.4. *Sia f una funzione continua e differenziabile definita su un aperto Ω di $\mathbb{R}^n$. Allora la sua variazione totale può essere espressa anche dalla seguente formula:*

$$V(f, \Omega) = \int_\Omega |\nabla f(x)| \, \mathrm{d}x. \tag{1.4}$$

Qui $|\cdot|$ rappresenta la norma euclidea.

1.2 Rimozione del rumore tramite variazione totale

In generale le tecniche di rimozione del rumore che operano tramite algoritmi di minimizzazione della variazione totale della funzione che rappresenta un determinato segnale seguono tutte lo stesso principio di fondo: i segnali con un numero elevato di dettagli isolati e discontinui hanno un'alta variazione totale; in altre parole, l'integrale della norma del gradiente associato al segnale è alto. Secondo questo principio, riducendo la variazione totale del segnale affetto da un qualche tipo di rumore si dovrebbero rimuovere i dettagli indesiderati e allo stesso tempo si dovrebbero preservare dettagli importanti come i bordi. D'altra parte però è altrettanto importante non

ridurre troppo la variazione del segnale: si potrebbero scambiare per rumore anche altri dettagli significativi. È per questo che di solito un buon algoritmo di rimozione del rumore tramite variazione totale ha al suo interno anche un secondo termine che, regolato dal già accennato parametro di split λ, permette di presentare determinati dettagli ritenuti importanti. Questo termine è spesso rappresentato da una qualche norma applicata al residuo, ovvero alla differenza tra l'immagine originale e l'immagine pulita (in altre parole, il rumore rimosso). Il problema di capire cosa è struttura e cosa è texture è un problema da non sottovalutare, data la sua forte soggettività. Come vedremo più avanti, lo spazio associato alla norma scelta giocherà un ruolo cruciale sull'efficacia dell'algoritmo rispetto alle varie tipologie di rumore.

Da qui in poi denoteremo con f la funzione rappresentante il segnale. È tipico e ragionevole considerare Ω come un insieme chiuso e limitato. Di solito, infatti, è un rettangolo (basta pensare al segnale come ad un'immagine). Com'è altrettanto ragionevole considerare f limitata in Ω. Di conseguenza, $f \in L^\infty(\Omega)$. Dato che Ω è limitato, vale anche che $f \in L^2(\Omega)$. Sia $BV(\Omega)$ l'insieme di tutte le funzioni con variazione totale limitata in Ω (i.e. $V(f, \Omega) < +\infty$).

Tutti gli algoritmi di rimozione del rumore che affronteremo in questo lavoro sono della forma

$$\inf \left(\int |Du| + \lambda \|v\|_X^p \right), \tag{1.5}$$

dove X rappresenta un certo insieme (per esempio $L^2(\Omega)$) e l'inf è da cercare tra tutte le coppie di funzioni $(u, v) \in (BV(\Omega) \times X)$ tali che $u + v = f$. Il parametro p rappresenta solo un esponente naturale mentre λ è il famoso *splitting parameter*, ovvero il parametro che regola quanto peso debba avere la somiglianza tra l'immagine pulita e l'immagine originale.

Questi algoritmi hanno lo scopo di separare l'immagine originale, rappresentata dalla funzione f, in due immagini diverse, rappresentate dalle funzioni u e v, la prima appartenente all'insieme delle funzioni a variazione totale limitata e contenente le informazioni significative dell'immagine (la parte pulita, chiamata parte struttura), la seconda appartentente ad uno spazio appositamente scelto e contenente i dettagli ritenuti insignificativi (la parte texture). Come si può facilmente evincere dalla formulazione generica,

il metodo si basa sulla minimizzazione di un funzionale che tiene conto sia della variazione totale della parte struttura di f, sia della norma della funzione texture v, che rappresenta il rumore rimosso. L'influenza di tale controllo sulla norma dei residui viene stabilita tramite il parametro λ. Questo parametro gioca un ruolo cruciale nell'implementazione del metodo: se $\lambda = 0$ l'unica richiesta nella decomposizione è la minimizzazione della variazione totale della funzione struttura. Questo vuol dire che il metodo ci restituirà un'immagine esageratamente ripulita, priva di qualunque dettaglio. Al contrario, se $\lambda \longrightarrow +\infty$ la parte di controllo della norma del rumore rimosso avrà sempre più peso nella minimizzazione, scavalcando la variazione totale. Il risultato prodotto sarà un'immagine non pulita, molto simile all'immagine originale.

Tuttavia, trovare il giusto valore del parametro λ non è affatto semplice. In questo lavoro proporremmo anche un metodo di ricerca automatico per questo parametro, che chiameremo semplicemente *algoritmo di correlazione* (a volte *algoritmo di selezione del parametro* λ). Come si può dedurre dal nome, il principio base che sta dietro quest'algoritmo è proprio la correlazione statistica. In particolare, la correlazione tra le due immagini di output. Il principio di base è:

Teorema 1.2.1 (Principio di correlazione)**.** *La parte struttura e la parte texture di un'immagine non sono correlate.*

L'algoritmo di correlazione calcola la soluzione del problema di decomposizione (1.5) partendo da un valore iniziale di λ. Poi via via continua a calcolare altre soluzioni al variare dello stesso λ. Per ogni coppia di soluzioni, ne calcola la correlazione (seguendo algoritmi numerici che spiegheremo più avanti). Il parametro λ ottimale viene scelto tra quelli che minimizzano tale correlazione.

1.3 Decomposizione di immagini astronomiche

Le immagini astronomiche vengono acquisite da sensori CCD posti al fuoco di sistemi ottici complessi di cui il telescopio (generalmente costruito con specchi a simmetria conica) costituisce il principale elemento. Un sensore

CCD è, in pratica, un contatore proporzionale di elettroni di conduzione generati assorbendo i fotoni incidenti su di esso. Sebbene il sistema ottico di un telescopio e dei suoi strumenti sia molto più complesso di una comune macchina fotografica o videocamera, le immagini astronomiche sono essenzialmente delle fotografie digitali del cielo tali da permettere un'analisi *fisica* in termini di *flusso* e *posizione* degli oggetti celesti.

Una sostanziale differenza rispetto alle comuni fotografie è data proprio dal fatto che ai valori (conteggi) registrati nei pixel di immagini astronomiche deve poter essere assegnato un valore in termini di fotoni ricevuti dalla corrispondente regione di cielo. Per questo motivo le immagini sono di fatto in scala di grigi ma con un range dinamico più elevato della comune scala [0-255]: generalmente si tratta di immagini a 16bit (equivalente a 65536 livelli di grigio), o comunque mai inferiori a 12bit (4096 livelli).

Analizzando un'immagine astronomica ci si rende subito conto che il rumore è una componente per nulla trascurabile. Spesso è talmente presente da coprire intere sorgenti luminose. Come visto in precedenza il segnale è dato dal numero di fotoni provenienti dalla sorgente che incidono sul singolo pixel. Per contro, il rumore ha diverse componenti:

- il rumore Poissoniano del segnale stesso
- il rumore dovuto a fotoni non provenienti dalla sorgente ma da luce spuria raccolta dal telescopio o da emissioni atmosferiche (il cosidetto "*sky-background*")
- il rumore elettronico di tipo shut-noise dovuto alla lettura del sensore ("*readout-noise*")
- il rumore termico, ovvero elettroni di conduzione generati dall'agitazione termica all'interno del reticolo cristallino ("*dark-current*")

In gran parte dei casi il termine di rumore dominante nei pixel non occupati da sorgenti è dovuto allo *sky-background* che, con buona approssimazione, può essere considerato Gaussiano. Un esempio è mostrato in Fig.1.2. Come si può vedere, i valori dei pixel ("conteggi") delle sorgenti più deboli vanno a confondersi con il rumore centrato attorno al valore zero. Di fatto,

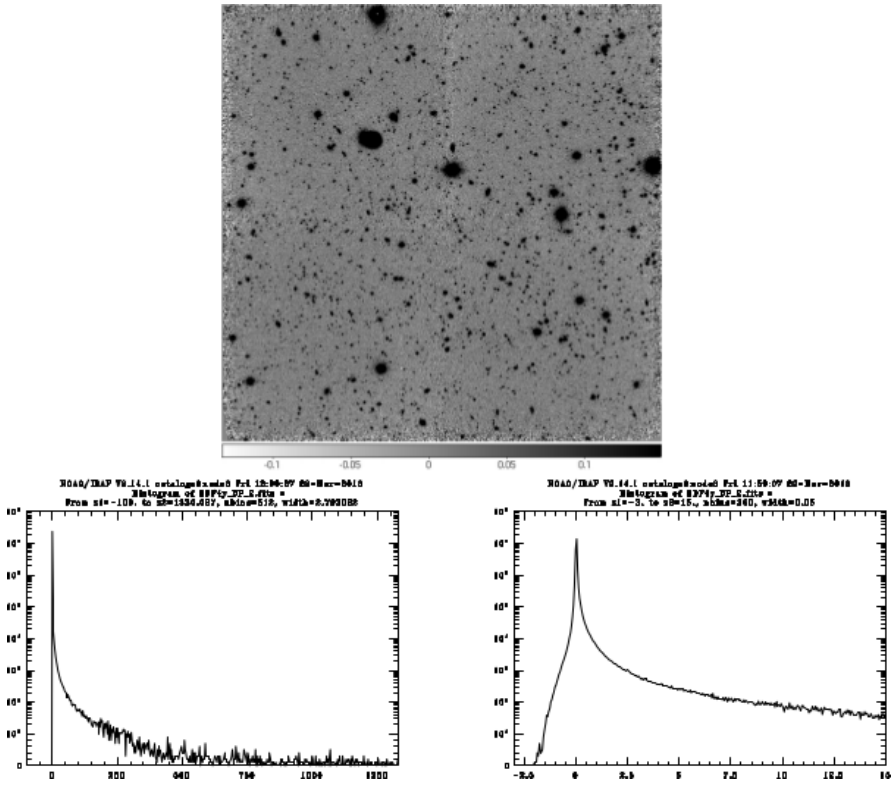


Figura 1.2: In alto: esempio di immagine astronomica (infrarosso, tempo di esposizione di 16 ore). Nell'immagine è presente un alto livello di rumore, e la separazione tra sorgenti e rumore è tutt'altro che ovvia, come mostrato dalle figure in basso: istogramma dei valori dei pixel (a sinistra) e ingrandimento dell'istogramma attorno a 0 (destra).

l'analisi scientifica delle immagini è limitata proprio dal rumore che impedisce di individuare le sorgenti più deboli o un maggior numero di dettagli nella distribuzione di luce di quelle più brillanti. La necessità di ridurre gli effetti del rumore nelle osservazioni ha da sempre spinto all'utilizzo di tempi di esposizione più lunghi e alla costruzione di telescopi più grandi e strumenti più sensibili.

Per questo motivo sarebbe di enorme interesse per l'astrofisica la possibilità di applicare in modo affidabile tecniche software di decomposizione structure-texture che rimuovano (o riducano) la presenza di rumore.

Capitolo 2

Algoritmi di decomposizione e di selezione parametri

In questo capitolo studieremo tre problemi di minimizzazione associati alla decomposizione struttura-texture dell'immagine data. Tali problemi hanno tutti la stessa forma, espressa in (1.5), differiscono l'uno dall'altro solo per la scelta dello spazio per la parte texture e, di conseguenza, della norma da utilizzare. In questo capitolo presenteremo i problemi associati alla norma di L^2, alla norma di L^1 e alla norma di G (vedi [1]). Li analizzeremo sia nel continuo sia nel discreto, dopo averli opportunatamente tradotti. Prenderemo in considerazione tre diversi algoritmi risolutivi, uno per ciascun problema, e li implementeremo tutti e tre in un unico programma scritto in C++, al quale verrà dato ampio risalto nell'ultima parte del capitolo.

2.1 I tre problemi di minimizzazione

Ricordiamo le ipotesi fatte su f nella Sez. 1.2. Il primo algoritmo che presentiamo è noto con il nome di *ROF Method* (dalle iniziali dei cognomi degli autori, Rudolf, Osher e Fatemi), presentato per la prima volta in [20], anche conosciuto con il nome di $TV - L^2$. Questo algoritmo scompone l'immagine di input, f, in una componente struttura $u \in BV$ (*Bounded Variation*) e in una componente texture $v \in L^2$. La norma scelta è quindi quella di L^2. Si

presenta nella seguente forma

$$\begin{aligned} &TV - L^2 \text{ } Decomposition \text{ } Problem \\ &\inf_{(u,v)\in(BV\times L^2) \ | \ u+v=f} \left(\int |Du| + \lambda \left\| v \right\|_{L^2}^2 \right). \end{aligned} \tag{2.1}$$

Come osservato anche dagli autori la scelta di L^2 fa di quest'algoritmo una sorta di decompositore universale, in grado di rimuovere la maggior parte dei tipi di rumori esistenti. Funziona particolarmente bene su texture con pattern oscillatori anche se, come vedremo in seguito, il prossimo algoritmo riesce ad ottenere risultati migliori su questo tipo di texture.

In [16], Meyer propose un altro algoritmo, incentrato su un particolare spazio di Banach, G. Il problema si presenta nella solita forma,

$$\begin{aligned} &TV - G \text{ } Decomposition \text{ } Problem \\ &\inf_{(u,v)\in(BV\times G) \ | \ u+v=f} \left(\int |Du| + \lambda \left\| v \right\|_{G} \right), \end{aligned} \tag{2.2}$$

con l'unica differenza che al posto di L^2 questa volta troviamo lo spazio G. Pensato appositamente per contenere segnali con grandi oscillazioni (come molti segnali puramente rumorosi), lo spazio di Banach G (e la relativa norma associata) è definito dalla seguente relazione:

Definizione 2.1.1. G è uno spazio di Banach composto da distribuzioni f che possono essere scritte nella forma

$$f = \partial_1 g_1 + \partial_2 g_2 = div(g), \tag{2.3}$$

dove $g_1, g_2 \in L^\infty$. La norma associata allo spazio G, $\| \cdot \|_G$, è definita dalla relazione

$$\|v\|_G = \inf \left\{ \|g\|_{L^\infty} \mid v = div(g), \ g = (g_1, g_2), \ g_1, g_2 \in L^\infty \right\}, \tag{2.4}$$

dove $|g(x)| = \sqrt{(|g_1|^2 + |g_2|^2)\,(x)}$.

Normalmente, una funzione con un pattern caratterizzato da oscillazioni molto ampie ha quasi sempre una norma grande rispetto alle altre funzioni con oscillazioni di minore ampiezza, anche se alla stessa frequenza. La norma $\| \cdot \|_G$ è stata quindi studiata per essere minima, in modo da catturare come segnali rumorosi anche segnali con grandi sbalzi tra i vari pixels, a patto

che siano di natura oscillatoria. Questa tecnica si adatta benissimo alle esigenze dell'astronomia: infatti molto spesso i disturbi legati alle immagini astronomiche presentano esattamente queste caratteristiche. Non a caso, come vedremo nel capitolo dedicato alle simulazioni numeriche, i migliori risultati si otterranno proprio con questo algoritmo.

In [2] e [18] gli autori hanno proposto di sostituire la norma L^2 del modello di ROF con la norma L^1. Il risultato è il seguente algoritmo

$$\begin{array}{l} TV - L^1 \ \textit{Decomposition Problem} \\ \inf_{(u,v)\in(BV\times L^1) \ | \ u+v=f} \left(\int |Du| + \lambda \left\| v \right\|_{L^1} \right). \end{array} \tag{2.5}$$

In uno dei suoi lavori ([18]) Nikolova ha dimostrato che cercare la parte texture di un'immagine all'interno di uno spazio come L^1 è una procedura particolarmente indicata per rimuovere rumori di tipo salt & pepper.

2.2 Discretizzazione

D'ora in avanti considereremo soltanto il caso discreto. L'immagine si presenterà quindi come un array bidimensionale $N \times M$. Denotiamo con X lo spazio euclideo $\mathbb{R}^{N\times M}$ e con Y lo spazio equivalente a $X \times X$. Sia $(u,v)_{L^2} = \sum_i \sum_j u_{i,j} v_{i,j}$ il prodotto scalare associato a X (discretizzazione del prodotto scalare di L^2) e $\|u\|_{L^2} = \sqrt{(u,u)_{L^2}}$ la discretizzazione della norma L^2. Definiamo anche $\|u\|_{L^1} = \sum_i \sum_j |u_{i,j}|$. Per definire una versione discreta della variazione totale di una funzione dobbiamo prima introdurre la versione discretizzata dell'operatore gradiente; se $u \in X$, allora ∇u è un vettore appartenente ad Y le cui componenti $(\nabla u)_{i,j} = \left((\nabla u)^1_{i,j}, (\nabla u)^2_{i,j} \right)$ sono date dalle seguenti espressioni:

$$(\nabla u)^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{se } i < N, \\ 0 & \text{se } i = N, \end{cases} \tag{2.6}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{se } j < M. \\ 0 & \text{se } j = M. \end{cases} \tag{2.7}$$

Definiamo quindi la variazione totale discretizzata di u con l'espressione

$$J(u) = \sum_{i=1}^{N} \sum_{j=1}^{M} \left| (\nabla u)_{i,j} \right|. \tag{2.8}$$

Introduciamo in questa sezione anche la versione discretizzata dell'operatore divergenza e dello spazio di Banach G, ci saranno utili in diversi punti. Sia $p \in Y$; la divergenza discretizzata proposta in [1] è data dalla seguente formula:

$$(div(p))_{i,j} = \begin{cases} p^1_{i,j} - p^1_{i-1,j} & \text{se } 1 < i < N \\ p^1_{i,j} & \text{se } i = 1 \\ -p^1_{i-1,j} & \text{se } i = N \end{cases} + \begin{cases} p^2_{i,j} - p^2_{i,j-1} & \text{se } 1 < j < M. \\ p^2_{i,j} & \text{se } j = M. \\ -p^2_{i,j-1} & \text{se } j = M. \end{cases} \tag{2.9}$$

Lo spazio G e la sua norma discretizzata associata sono invece descritti dalla seguente definizione:

Definizione 2.2.1.

$$G = \{v \in X \mid \exists g \in Y \text{ tale che } v = div(g)\}.$$

Se $v \in G$:

$$\|v\|_G = \inf \left\{ \|g\|_{L^\infty} \mid v = div(g), g = (g^1, g^2) \in Y \right\},$$

dove $\|g\|_{L^\infty} = \max_{i,j} |g_{i,j}|$.

Sia $\mu \in \mathbb{R}$. Definiamo il seguente sottospazio di G:

$$G_\mu = \{v \in G \mid \|v\|_G \leq \mu\}.$$

In ultimo presentiamo una proposizione, dimostrata in [4], che ci sarà utile più avanti;

Proposizione 2.2.2. *Lo spazio G equivale allo spazio*

$$X_0 = \left\{ v \in X \ : \ \sum_{i=1}^{N} \sum_{j=1}^{M} v_{i,j} = 0 \right\}.$$

2.3 L'algoritmo di proiezione di Chambolle

Nell'articolo [6], l'autore ha proposto un interessante algoritmo nonlineare di proiezione per risolvere il problema ROF (2.1) (il $TV - L^2$). Il problema discretizzato preso in considerazione è

$$\inf_{u \in X} \left(J(u) + \frac{1}{2\lambda} \|f - u\|^2_{L^2} \right). \tag{2.10}$$

Il seguente teorema, dimostrato in [6], mostra una via da seguire per arrivare ad una soluzione del problema (2.10)

Teorema 2.3.1. *La soluzione di* (2.10) *è data da* $u = f - P_{G_\lambda}(f)$*, dove* P *rappresenta la proiezione ortogonale di* f *su* G_λ*.*

Tutto sta quindi nel cercare un algoritmo efficace per calcolare $P_{G_\lambda}(f)$. Sempre in [6] viene proposto un algoritmo iterativo per il calcolo della proiezione. Si dimostra che è sufficiente cercare $p \in Y$ soluzione del problema

$$\min \left\{ \|\lambda div(p) - f\|_{L^2}^2 \ : \ p/|p_{i,j}| \leq 1 \quad \forall i = \ 1, \dots, N, \ \forall j = 1, \dots, M \right\}$$

tramite il seguente algoritmo di iterazione a punto fisso

$$\begin{aligned} &p_{i,j}^0 = 0. \\ &p_{i,j}^{n+1} = \frac{p_{i,j}^n + \tau(\nabla(div(p^n) - f/\lambda))_{i,j}}{1 + \tau\left|(\nabla(div(p^n) - f/\lambda))_{i,j}\right|}, \qquad n \in \mathbb{N}, \end{aligned} \tag{2.11}$$

dove ricordiamo che con la notazione $(p)_{i,j}$, si intende il vettore $\left(p_{i,j}^1, p_{i,j}^2\right)$. Condizione sufficiente affinché l'algoritmo converga ad una soluzione è che il parametro τ rimanga minore di $\frac{1}{8}$; in tal caso si dimostra che per $n \longrightarrow +\infty$

$$\lambda div(p^n) \longrightarrow P_{G_\lambda}(f).$$

2.4 L'algoritmo di Aujol-Aubert-Blanc-Féraud-Chambolle (A^2BC)

Inspirati dal lavoro di Chambolle [6] e dai risultati numerici mostrati in [21] gli autori di [4, 5] hanno proposto un algoritmo molto versatile ed efficiente per risolvere il problema di Meyer (2.2) (il problema $TV - G$). In [4] viene proposta la seguente discretizzazione del problema:

$$\inf_{(u,v) \in X \times G_\mu} \left(J(u) + \frac{1}{2\alpha} \|f - u - v\|_{L^2}^2 \right), \tag{2.12}$$

dove $G_\mu = \{v \in G \mid \|v\|_G \leq \mu\}$ e $\|v\|_G$ rappresenta la norma descritta in (2.4). Osserviamo che la richiesta $u + v = f$ è sostituita da un controllo sulla norma della differenza fra f e $u + v$. Il parametro α si occupa proprio di

regolare l'importanza di tale somiglianza. Più è piccolo e più la somiglianza richiesta è stretta. Il parametro μ invece controlla la norma (nel senso definito da (2.4)) della parte texture v; in altre parole, controlla la quantità di informazioni contenute in v. È il nostro *splitting parameter*. Vedremo più avanti come scegliere in maniera ottimale il suo valore.

L'algoritmo proposto in [4], conosciuto con il nome di A^2BC, si basa sull'algoritmo di proiezione di Chambolle per la soluzione del problema *ROF* (2.10). Per dare almeno un'idea del loro approccio, introduciamo una proposizione (dimostrata in [12]):

Proposizione 2.4.1. *Sia J^* l'operatore aggiunto di J definita in* (2.8)*. Sia $G_1 = \{v \in G \mid \|v\|_G \leq 1\}$. Allora*

$$J^*(v) = \begin{cases} 0 & \text{se } v \in G_1 \\ +\infty & \text{altrimenti} \end{cases}$$

In base a quanto appena detto, possiamo riscrivere (2.12) nella forma

$$\inf_{(u,v)\in X\times X} \left(J(u) + \frac{1}{2\alpha}\|f-u-v\|_{L^2}^2 + J\left(\frac{v}{\mu}\right)\right). \tag{2.13}$$

Qui è più evidente il ruolo simmetrico giocato da u e da v. Partendo da questa osservazione, gli autori di [4] hanno pensato di costruire il loro algoritmo dividendo il problema principale. Precisamente:

1. v fissata, si cerca una soluzione di

$$\inf_{u\in X} \left(J(u) + \frac{1}{2\alpha}\|f-u-v\|_{L^2}^2)\right), \tag{2.14}$$

2. u fissata, si cerca una soluzione di

$$\inf_{v\in G_\mu} \|f-u-v\|_{L^2}^2. \tag{2.15}$$

Dal teorema 2.3.1 sappiamo che la soluzione di 2.14 è data da

$$\hat{u} = f - v - P_{G_\alpha}(f-v).$$

Come dimostrato nell'articolo [4], la soluzione di 2.15 è data da

$$\hat{v} = P_{G_\mu}(f-u).$$

Algoritmo A^2BC:

1. Inizializzazione:
$$u_0 = \ v_0 \ = 0.$$

2. Iterazioni:
$$v_{n+1} = P_{G_\mu}(f - u_n),$$
$$u_{n+1} = f - v_{n+1} - P_{G_\alpha}(f - v_{n+1}).$$

3. Criterio di arresto:
$$\max\left(|u_{n+1} - u_n| \ , \ |v_{n+1} - v_n|\right) \leq \epsilon$$

2.5 L'algoritmo di Soft Thresholding

In [8] viene proposto un algoritmo per risolvere il seguente problema di minimizzazione discretizzato:

$$\inf_{(u,v)\in X\times X}\left(J(u) + \frac{1}{2\alpha}\|f - u - v\|_{L^2}^2 + \lambda\|v\|_{L^1}\right). \tag{2.16}$$

Anche qui l'ipotesi $f = u + v$ viene abbandonata a favore di una meno restrittiva richiesta di somiglianza tra i due elementi. A parte questo, il modello si comporta esattamente come descritto nella Sez. 1.2. Il parametro α regola l'importanza della somiglianza tra f e $u+v$, mentre il parametro λ divide la parte struttura dalla parte texture. Con questa formulazione, più λ è piccolo e più si tendono a considerare dettagli come texture.

Prima di risolverlo riportiamo una proposizione, sempre contenuta in [8]:

Proposizione 2.5.1. *Sia $\beta \in \mathbb{R}^+$. La soluzione del problema*

$$\inf_{v\in X}\left\{\frac{1}{2\beta}\|g - v\|_{L^2}^2 \ + \ \|v\|_{L^1}\right\}$$

è data da

$$v_{i,j} = \begin{cases} g_{i,j} - \beta & se\ g_{i,j} \geq \beta \\ 0 & se\ |g_{i,j}| \leq \beta \\ g_{i,j} + \beta & se\ g_{i,j} \leq -\beta \end{cases}$$

Scriveremo $v = ST(g, \beta)$ $(Soft - Thresholding)$ per indicare la soluzione calcolata secondo 2.5.1.

Torniamo adesso al problema (2.16). Anche qui, il modus operandi ricalca molto quanto visto nella sezione precedente. In questo caso sappiamo (sempre da [8]) che (2.16) è un funzionale convesso. Anche qui possiamo dividere il problema, fissando prima un'incognita e poi l'altra. L'iterazione delle soluzioni per i due problemi porterà quindi all'unica soluzione. Per un approccio più generico a questo tipo di problemi si invita il lettore a consultare [9, 10].

Fissando v, si ottiene il seguente problema:

$$\inf_{u \in X} \left(J(u) + \frac{1}{2\alpha} \|f - u - v\|_{L^2}^2 \right), \tag{2.17}$$

mentre fissando u otteniamo:

$$\inf_{v \in X} \left\{ \frac{1}{2\alpha} \|f - u - v\|_{L^2}^2 \ + \ \|v\|_{L^1} \right\}. \tag{2.18}$$

Dal teorema 2.3.1 sappiamo che la soluzione di (2.17) è data da $\hat{u} = f - u - P_{G_\alpha}(f - v)$, mentre la proposizione 2.5.1 ci dà la soluzione del problema (2.18).

Algoritmo $Soft - Thresholding$:

1. Inizializzazione:
$$u_0 = \ v_0 \ = 0.$$

2. Iterazioni:
$$v_{n+1} = ST(f - u_n, \alpha\lambda),$$
$$u_{n+1} = f - v_{n+1} - P_{G_\alpha}(f - v_{n+1}).$$

3. Criterio di arresto:
$$\max\left(|u_{n+1} - u_n| \ , \ |v_{n+1} - v_n|\right) \leq \epsilon.$$

2.6 L'algoritmo di correlazione: selezione automatica del parametro di splitting

Nelle precedenti sezioni abbiamo presentato le versioni discretizzate dei problemi conosciuti come $TV - L^2$, $TV - G$, $TV - L^1$. Abbiamo visto che, in generale, tutti quanti seguono il modello proposto nella Sez. 1.2, ovvero in ognuno di loro è presente un parametro di *splitting* responsabile del peso che si intende attribuire ai dettagli dell'immagine da scomporre. In effetti trovare il giusto valore per questo parametro non è sempre molto semplice: la divisione tra dettaglio significativo e texture è un concetto molto soggettivo, dipendente da molti dettagli (come ad esempio le dimensioni degli oggetti ritratti, la loro prospettiva ecc.).

Lo scopo di questa sezione è quello di presentare un algoritmo che permetta di stabilire in maniera automatica qual'è il miglior valore per il suddetto parametro, basandosi unicamente sulle proprietà dell'immagine che sta analizzando. Il nostro approccio segue il lavoro di Mrazek-Navara ([17]) e di Aujol ([1]). Il metodo si basa su un criterio di correlazione e non richiede alcuna conoscenza a priori di particolari proprietà della parte texture dell'immagine (come la varianza nel caso di rumore).

La generica decomposizione affrontata in questo lavoro è data dalla seguente espressione:

$$\begin{aligned} &E_{Struttura}(u) + \lambda E_{Texture}(v), \\ &f = u + v, \end{aligned} \tag{2.19}$$

dove u e v minimizzano la somma dei funzionali. Il nostro obiettivo è quello di trovare il miglior bilanciamento tra i due termini della somma (in altre parole, il miglior valore di λ) tale che venga prodotta una decomposizione sufficientemente significativa dell'immagine in questione.

Prima di descrivere l'algoritmo, premettiamo qualche definizione.

Definizione 2.6.1 (Valore atteso, varianza e covarianza discreti)**.** Siano $q, r \in X$. Allora definiamo:

· Valore atteso:

$$\bar{q} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} q_{i,j}$$

· Varianza:

$$V(q) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (q_{i,j} - \bar{q})^2$$

· Covarianza:

$$Cov(q, r) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (q_{i,j} - \bar{q})(r_{i,j} - \bar{r})$$

Sotto le ipotesi della precedente definizione presentiamo in ultimo la versione discretizzata del coefficiente di correlazione:

$$corr(q, r) := \frac{Cov(q, r)}{\sqrt{V(q)V(r)}}. \tag{2.20}$$

Ovviamente tale coefficiente ha senso se e solo se $V(q)V(r) > 0$. In tal caso, vuol dire che una delle due soluzioni equivale ad una costante. In tal caso, per scartare tale ipotesi, in fase di implementazione si può dare un valore fittizio alla correlazione calcolata. Vedremo più avanti che l'unica soluzione costante possibile è la funzione costante equivalente al valore medio della funzione rappresentante l'immagine. Grazie alla disuguaglianza di Cauchy-Schwarz non è difficile rendersi conto che $Cov(q, r) \leq \sqrt{V(q)V(r)}$, quindi che

$|corr(q,r)| \leq 1$. Il massimo valore 1 (*completa correlazione*) è raggiunto solo quando si calcola tale coefficiente per segnali identici. L'antipodale valore -1 invece (*completa anticorrelazione*) si raggiunge comunque calcolando la correlazione di segnali molto simili tra loro, ma legati da una qualche relazione basata su un moltiplicare negativo. Quando la correlazione calcolata è 0, ci troviamo davanti a due segnali completamente *scorrelati*. Essere scorrelati è una condizione necessaria, ma non sufficiente, per avere indipendenza statistica. Nell'ambito della teoria dei segnali spesso la scorrelazione implica anche che i segnali sono stati prodotti da diversi modelli di generatori, come per esempio un segnale contenente un'informazione e un segnale composto da rumore gaussiano.

L'algoritmo di selezione del parametro di splitting che presentiamo si basa su un'unica assunzione, ovvero che *parte struttura e parte texture di una stessa immagine non sono correlate tra di loro*. In parole povere, l'algoritmo valuta la correlazione tra diverse coppie di soluzioni calcolate al variare del valore di λ. L'ideale sarebbe trovare una coppia per cui si riesca a calcolare una correlazione pari a 0. Nella pratica questo non succede mai. Si opterà allora per cercare semplicemente il valore di λ cui è associata la coppia di soluzioni che genera la correlazione più bassa possibile.

Consideriamo ora il caso della scelta del parametro λ per la soluzione del problema *ROF* (2.10). Sia (u_λ, v_λ) una coppia di soluzioni per il suddetto problema (ottenuta tramite l'algoritmo di proiezione di Chambolle, vedi Sez. 2.3). Un semplice risultato, contenuto in [16], ci assicura che $Cov(u_\lambda, v_\lambda) \geq 0$ per ogni scelta di $\lambda > 0$. In questo caso non dovremmo quindi preoccuparci di correlazioni negative. Definiamo λ^* *parametro ottimale* per il problema (2.10) la soluzione di:

$$\lambda^* = argmin_\lambda \left(corr(u_\lambda, v_\lambda) \right). \tag{2.21}$$

Nella pratica, si definisce un vettore con un certo numero di valori possibili per λ (tipicamente decrescenti) e si calcola la soluzione di (2.10) associata ad ogni valore scelto. Per ogni coppia di soluzioni, si calcola la rispettiva correlazione e si salva il risultato in un altro vettore. Ovviamente, nel caso dell'algoritmo di Chambolle u_λ è la soluzione calcolata mentre $v_\lambda = f - u_\lambda$.

Il valore di λ associato al minimo del vettore composto dalle correlazioni sarà il nostro parametro ottimale.

Anche l'ottimizzazione del parametro di splitting per il problema $TV-L^1$ (2.16) segue le stesse dinamiche. Si itera l'algoritmo di Soft-Thresholding sullo stesso vettore di parametri λ definito prima e si calcolano allo stesso modo le correlazioni tra le varie coppie di soluzioni ottenute. L'unica cosa cui dobbiamo fare attenzione è che in questo caso non esiste nessun teorema che ci assicura la non negatività delle correlazioni calcolate. Di conseguenza non cercheremo più il minimo del vettore delle correlazioni, ma il valore più vicino allo 0.

Per quanto riguarda il problema di Meyer (2.12), il discorso è praticamente analogo. Un primo approccio potrebbe essere quello di calcolare il parametro ottimale sfruttando direttamente diverse iterazioni dell'algoritmo A^2BC. Approccio scartato per via dei tempi computazionali: come approfondiremo nel capitolo dedicato alle simulazioni e ai risultati numerici, l'algoritmo A^2BC ha tempi computazionali decisamente lunghi rispetto all'algoritmo risolutivo per ROF. Per affrontare il problema, gli autori di [1] hanno proposto una strada alternativa: calcolare la il parametro ottimale λ sfruttando l'algoritmo di proiezione di Chambolle. Ovviamente per poter funzionare tale approccio richiede almeno la conoscenza di che tipo di relazione c'è tra il parametro λ di (2.10) e il parametro μ di (2.12). Argomento che affronteremo nella prossima sottosezione.

2.6.1 La relazione tra μ e λ

La norma G è stata introdotta da Meyer in [16] per analizzare le proprietà matematiche del modello ROF proposto in (2.10). Uno dei maggior risultati contenuti in quell'articolo è il seguente corollario (relativo al teorema (2.3.1)):

Corollario 2.6.2. *Sia u_λ soluzione di (2.1), $v_\lambda = f - u_\lambda$ e $\bar{f}$ il valore medio di f. Allora:*

1. *Se $\|f - \bar{f}\|_G \geq \lambda$ allora $\|v_\lambda\|_G = \lambda$;*
2. *Se $\|f - \bar{f}\|_G \leq \lambda$ allora $u_\lambda = \bar{f}$.*

Come possiamo quindi vedere, il comportamento del modello ROF è strettamente legato alla norma G dell'immagine f. Enunciamo e dimostriamo il seguente lemma:

Lemma 2.6.3. *Il parametro λ calcolato tramite l'algoritmo di selezione presentato nella Sez. 2.6 è tale che*

$$\|v_\lambda\|_G = \lambda.$$

Dimostrazione. La dimostrazione si basa sui risultati del corollario 2.6.2. In pratica, ci basta dimostrare che il primo caso del corollario non si verifica mai (se λ viene scelto tramite l'algoritmo di correlazione). Sia quindi $\lambda \geq \|f - \bar{f}\|_G$. Per il corollario 2.6.2 abbiamo che $u_\lambda = \bar{f}$; nella nostra ottica, questa soluzione ha ben poca importanza. Vuol dire che l'algoritmo ha sfocato completamente l'immagine, uniformando il valore di tutti i suoi pixel al loro valor medio. Come avevamo già osservato un valore di λ troppo alto può rimuovere dettagli significativi scambiandoli per texture. Il corollario 2.6.2 ci dà una soglia. Per verificare l'impossibilità di tale scelta tramite il nostro algoritmo di correlazione basti pensare che $V(u_\lambda$ e $Cov(u_\lambda, v_\lambda)$ calcolati sotto queste ipotesi risultano entrambi nulli. Questo implica che il coefficiente di correlazione non può essere calcolato; una buona implementazione dell'algortimo, infatti, prevede questo caso assegnando un valore fittizio alla correlazione, solitamente 1, scelto in modo tale che esso non possa mai essere il valore minimo. Di conseguenza il λ ottimale verrà scelto tra tutti i $\lambda < \|f - \bar{f}\|_G$. Grazie alla seconda parte del corollario 2.6.2 abbiamo la tesi. □

Come abbiamo detto nell'introduzione al problema (2.2), l'idea di Meyer era quella di sostituire la norma di L^2 con la norma G, molto più adatta per catturare texture dominate da pattern oscillatori. Grazie al lemma appena dimostrato conosciamo anche il valore della norma G di v calcolata tramite l'algoritmo di Chambolle. Quindi, se effettuassimo una decomposizione secondo Meyer dell'immagine originale con il vincolo $\|v\|_G = \lambda$, dovremmo ottenere risultati migliori se paragonati alla stessa decomposizione secondo ROF, visto che la norma G risulta essere più adatta a catturare i pattern

oscillatori rispetto alla norma L^2. Effettivamente i risultati numerici danno ragione a questa affermazione (vedi capitolo successivo).

Il lemma appena dimostrato e le successive osservazioni ci portano quindi a considerare il problema (2.12) scegliendo

$$\mu = \lambda.$$

Con tale scelta del parametro stiamo quindi cercando parti texture tali che $\|v\|_G \leq \lambda$ (vedi descrizione del modello). In realtà, si può dimostrare che con questa scelta si ha sempre uguaglianza. Prima di farlo, ricordiamo il problema associato al modello A^2BC. Il funzionale da minimizzare è:

$$F(u,v) = J(u) + J^*\left(\frac{v}{\mu}\right) + \frac{1}{2\alpha}\|f-u-v\|_{L^2}^2. \tag{2.22}$$

Il seguente lemma di esistenza e unicità è dimostrato in [4]

Lemma 2.6.4. *Esiste un'unica coppia di funzioni* $(\hat{u},\hat{v}) \in X \times G_\mu$ *che minimizza il funzionale descritto in* (2.22) *in* $X \times X$.

Enunciamo e dimostriamo la seguente proposizione

Proposizione 2.6.5. *Sia* f *l'immagine originale,* $\bar{f}$ *il suo valore medio e* $(\hat{u},\hat{v}) \in X \times G_\mu$ *l'unica coppia di funzioni minimizzatrici per* (2.22). *Si ha che:*

1. *Se* $\|f-\bar{f}\|_G \leq \mu$ *allora* $\hat{v} = f - \bar{f}$;
2. *Se* $\|f-\bar{f}\|_G \geq \mu$ *allora* $\|\hat{v}\|_G = \mu$.

Dimostrazione. Essendo il risultato di una somma di componenti sempre non negative, possiamo affermare che anche $F(u,v) \geq 0 \quad \forall (u,v) \in X \times X$, essendo F il funzionale descritto in (2.22). Sia $\|f-\bar{f}\|_G \leq \mu$, i.e. $f - \bar{f} \in G_\mu$ (il fatto che $f-\bar{f} \in G$ è vero per costruzione, basti pensare alla proposizione 2.2.2). È facile dimostrare che la coppia $(\bar{f}, f-\bar{f}) \in X \times X$ minimizza il funzionale non negativo F; con un rapido calcolo si ottiene $F(\bar{f}, f-\bar{f}) = 0$ (somma di addendi nulli). Per tanto, grazie all'unicità della soluzione (garantita dalla proposizione 2.6.4), si dimostra la il primo punto della proposizione.

Sia ora $\|f - \bar{f}\|_G \geq \mu$. Consideriamo il seguente funzionale definito su $X \times X$:

$$H(u, v) = J(u) + \frac{1}{2\alpha}\|f - u - v\|_{L^2}^2.$$

H è un funzionale strettamente convesso. Di conseguenza, esisterà un unica coppia $(\tilde{u}, \tilde{v}) \in X \times G_\mu$ tale da minimizzarlo su $X \times G_\mu$. Ricordiamo che, esattamente come F, $H(\bar{f}, f - \bar{f}) = 0$. Consideriamo la funzione

$$g : \ t \in [0, 1] \ \rightarrow \|t\tilde{v} + (1 - t)(f - \bar{f})\|_G;$$

g è una funzione continua in $[0, 1]$. Inoltre si ha che $g(0) = \|f - \bar{f}\|_G \geq \mu$ e che $g(1) = \|\tilde{v}\|_G \leq \mu$ (per definizione). Ovvero, esisterà un certo $\tilde{t} \in [0, 1]$ tale che $g(\tilde{t}) = \mu$. Sia ora

$$\acute{v} = \tilde{t}\tilde{v} + (1 - \tilde{t})(f - \bar{f}) \qquad \acute{u} = \tilde{t}\tilde{v} + (1 - \tilde{t})\bar{f}.$$

Grazie alla convessità di H abbiamo che

$$H(\acute{u}, \acute{v}) \leq \tilde{t}H(\tilde{u}, \tilde{v}) + (\tilde{t})H(\bar{f}, f - \bar{f}) \leq H(\tilde{u}, \tilde{v}).$$

Ne deduciamo quindi che $(\acute{u}, \acute{v})$ è una coppia di funzioni minimizzatrici per H su $X \times G_\mu$. Dato che H e F coincidono su $X \times G_\mu$, deduciamo che $(\acute{u}, \acute{v})$ è una coppia di funzioni minimizzatrici anche per F, su tutto $X \times X$. Grazie al lemma di unicità 2.6.4, possiamo dunque concludere affermando che $(\acute{u}, \acute{v}) = (\bar{u}, \bar{v})$ e che $\|\hat{v}\|_G = \|\acute{v}\|_G = \mu$. □

Dal lemma 2.6.3 e dal corollario 2.6.2 sappiamo che $\|f - \bar{f}\|_G \geq \lambda$. Nella sottosezione precedente abbiamo considerato $\mu = \lambda$. Grazie alla proposizione appena dimostrata possiamo dedurre che $\hat{v}$, la componente v ottenuta mediante l'algoritmo A^2BC, è tale che $\|\hat{v}\|_G = \mu$. Questa nuova componente v risulta quindi avere la stessa norma G della controparte ottenuta mediante l'algoritmo di Chambolle per ROF, ma dato che la norma G è stata studiata con l'intento di dare maggior risalto a pattern di tipo oscillatorio rispetto alla norma L^2, questa nuova decomposizione produrrà risultati migliori.

Quest'analisi verrà confermata nel capitolo dedicato ai risultati numerici.

2.7 Il Programma

ATVD (*Astral Total Variation Decomposer*, è un programma scritto in C++ che implementa gli algoritmi visti nelle sezioni precedenti in un unico codice. Esso è costituito da (vedi Fig. 2.1) con un codice sorgente principale (`ATVD.cpp`), un codice sorgente per la libreria contenente le funzioni usate dal codice principale (`master.cpp` e il suo header `master.h`), un foglio per i parametri da passare al programma (`Parametri.txt`), uno script in bash utile a velocizzarne le esecuzioni (`ATVD`) e un Makefile. Di default vengono fornite anche tre cartelle, `FITS` (che contiene le immagini da analizzare, in formato `.fits`), `PROVE` (destinata a stipare i risultati del programma, si può cambiare tramite un opzione nel foglio dei parametri) e `REPORT` (cartella di destinazione per i report del programma, ovvero tutti gli output visionabili a schermo).

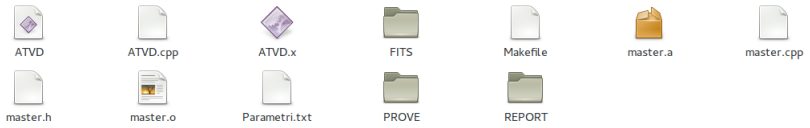


Figura 2.1: Le componenti di ATVD

2.7.1 Compilazione e Makefile

Il Makefile presente all'interno della cartella del programma permette di compilare l'eseguibile e la libreria associata. Le impostazioni di compilazione di default sono visibili attraverso il comando `make info`, che genera il risultato mostrato in Fig. 2.2.

```
master@master-desktop:~/ATVD$ make info
compilatore               g++
opzioni                   -O3 -funroll-loops
profiler                  -pg
parallelizzazione         -fopenmp
nome output               ATVD.x
master@master-desktop:~/ATVD$
```

Figura 2.2: `make info`

Come possiamo vedere, il programma ammette una parallelizzazione tramite `OpenMP`. Ricordiamo che, lavorando con una shell BASH, è possibile modificare il numero di threads riservati all'esecuzione del programma in parallelo attraverso il comando `export OMP_NUM_THREAD=n`, dove `n` sta per il numero di threads desiderato. Se si lavora sotto shell CSH o TCSH è possibile settare la stessa variabile attraverso il comando `setenv OMP_NUM_THREAD n`. Ovviamente è possibile modificare tutte le variabili mostrate da `make info` semplicemente modificando il Makefile.

L'opzione `make lib` crea la libreria `master.a` mentre `make atvd` crea l'eseguibile principale del programma. Le due opzioni possono essere chiamate insieme semplicemente scrivendo `make all`. Tutte le compilazioni vengono eseguite linkando anche la libreria `CFITSIO` (tramite l'opzione `-lcfitsio`), ovvero una libreria `C` (disponibile anche in `Fortran`) liberamente scaricabile dal sito `http://heasarc.gsfc.nasa.gov/fitsio/fitsio.html` che permette la gestione e la manipolazione delle immagini in formato `.fits`, formato utilizzato come standard per lavorare su immagini astronomiche, mettendo a disposizione funzioni di input e output (I/O).

Oltre alle opzioni di compilazione, il Makefile fornisce un'opzione `make clean` per rimuovere i file object e l'eseguibile generato e un'opzione `make conv` per convertire immagini di vario formato in immagini `.fits` (utilizzata principalmente per fare prove su immagini non astronomiche).

2.7.2 Esecuzione di ATVD

Prima di poter utilizzare il programma, è necessario compilare il foglio dei parametri (`Parametri.txt`, vedi Fig. 2.3). Nel foglio è possibile modi-

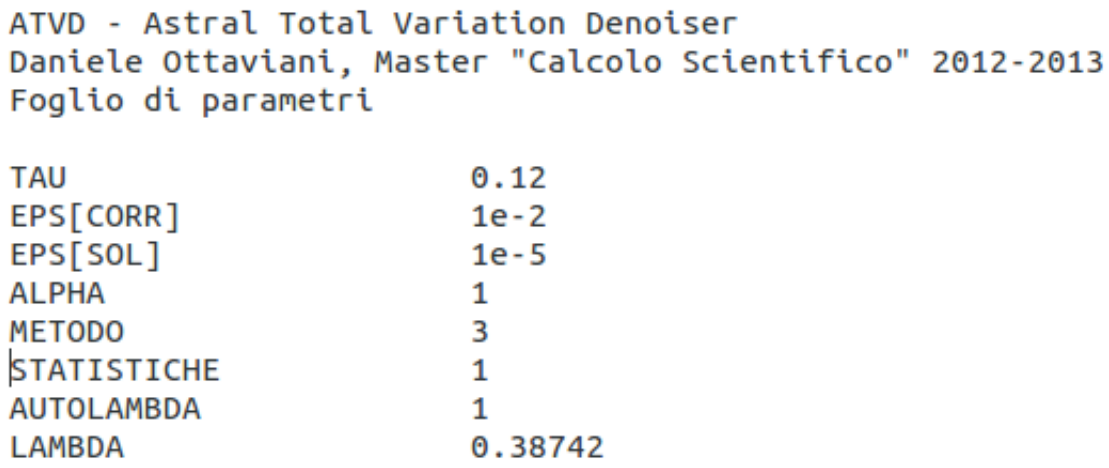


Figura 2.3: Foglio dei parametri: opzioni input

ficare i valori dei parametri associati ai vari metodi di decomposizione (τ, α e λ, vedi sezioni precedenti) e alcune caratteristiche legate all'esecuzione del programma, come le statistiche sul rumore rimosso. Andando con ordine abbiamo la possibilità di modificare:
`TAU`: Il passo dell'algoritmo scelto (τ).
`EPS`: La sensibilità dell'algoritmo di correlazione e del metodo (rispettivamente `EPS[CORR]` e `EPS[SOL]`). Ricordiamo infatti che tutti gli algoritmi scelti sono contrazioni che iterano verso il loro punto fisso.
`ALPHA`: Il valore del parametro α utilizzato dagli algoritmi $TV-L^1$ e $TV-G$ (sezioni 2.5 e 2.4).
`METODO`: Il metodo desiderato (1= $TV-L^1$, 2=$TV-L^2$, 3=$TV-G$).
`STATISTICHE`: Test statistici da produrre in output. È una variabile booleana, con 1 si attiva e con 0 si disattiva. Se attiva, produrrà nella cartella predestinata delle immagini in formato `.png` contenenti il grafico di correlazione relativo all'algoritmo di selezione del parametro λ, due grafici di autocorrelazione, due istogrammi e due $Q-Q$ Plot (con i quantili della distribuzione normale come termine di paragone) relativi al rumore estratto e al rumore originale (ove presente). Il file di output `Test_Gaussiani.Rout`

contiene inoltre i risultati di due test statistici effettuati sul rumore rimosso e sul rumore originale al fine di testarne indipendenza e gaussianità. I test scelti sono il test di Shapiro per la normalità dei dati e il test di Box-Pierce per l'indipendenza. Ognuno di questi risultati è ottenuto mediante l'utilizzo del software statistico `R` (`http://www.r-project.org/`), richiamato dal programma tramite un foglio di comando `Statistica.R` creato dal programma stesso.
`AUTOLAMBDA`: Variabile booleana riservata all'algoritmo per la ricerca automatica del parametro λ. Se vale 1 il programma ignorerà il valore di λ scritto nel foglio dei parametri e cercherà λ con l'algoritmo descritto nella Sez. 2.6. Se vale 0, il programma effettuerà un'unica decomposizione usando il valore di λ che gli è stato passato.
`LAMBDA`: Valore del parametro λ. Ignorato se AUTOLAMBDA=1.

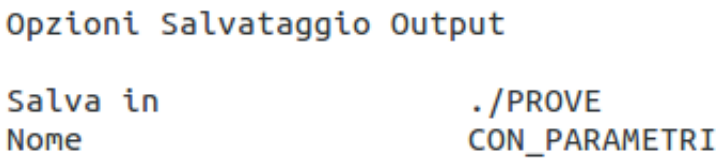


Figura 2.4: Foglio dei parametri: opzioni output

Nel foglio dei parametri è possibile anche gestire l'output di `ATVD` (vedi Fig. 2.4). In particolare, è possibile scegliere la cartella di destinazione dei risultati prodotti dal programma (per default è la cartella `PROVE`) e il nome della cartella contenente l'immagine decomposta (e tutti gli altri risultati che la riguardano).
Infatti, di default, `ATVD` salva i risultati prodotti in una cartella da lui creata. È da sottolineare il fatto che tali cartelle non vengono mai create facendo ricorso alla funzione `system`, ma tramite un'apposita funzione. Tramite il foglio dei parametri è possibile scegliere il nome di tale cartella. Precisamente:
`CON_PARAMETRI`: Genera una cartella con il nome dell'immagine scelta, un riassunto dei parametri e un identificativo ("tag"). Vedi Fig. 2.5
`SOLO_NOME`: Genera una cartella nominata solamente con il nome dell'immagine scelta.
Qualunque altro comando inserito verrà interpretato come il nome della car-

tella da creare (per esempio, scrivere `NEWDIR` equivale a dire al programma di creare una cartella di nome `NEWDIR` e di salvare tutto il suo output corrente in quella cartella. Ovviamente non è possibile creare cartelle chiamate `CON_PARAMETRI` o `SOLO_NOME`.

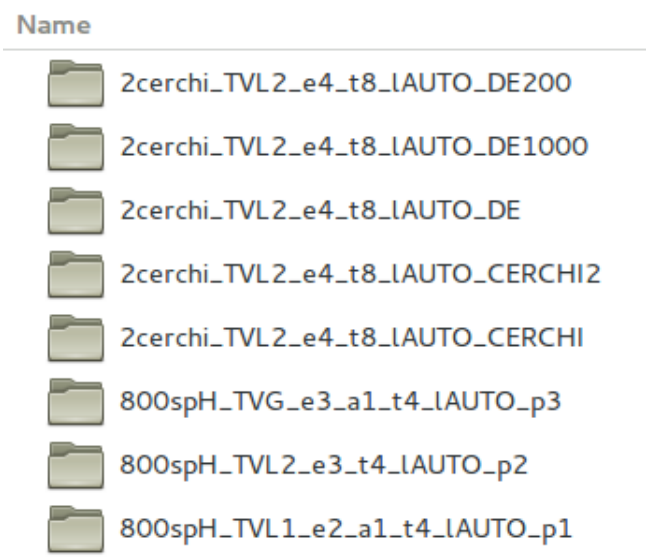


Figura 2.5: Esempi di cartelle create da ATVD. In ordine possiamo notare, separati da underscores, il nome dell'immagine originale, il metodo con cui è stata elaborata, la sensibilità scelta (`e`), α dove presente (`a`), il numero di threads coinvolti (`t`), il valore di λ (`l`, `AUTO` sta per automatico) e la tag.

Dopo aver scelto i parametri voluti, si può avviare `ATVD`. La corretta sintassi per l'esecuzione del programma è la seguente

```
./ATVD.x ./FITS/Image.fits IMAGE_NAME TAG ./FITS/Imageclean.fits
```

dove il primo elemento del vettore `argv` rappresenta il nome del programma, il secondo il `PATH` dell'immagine da analizzare, il terzo rappresenta il nome dell'immagine (solitamente è il `PATH` privato dell'indirizzo e dell'estensione), il quarto la "tag" (o identificativo dell'esecuzione). Il quinto parametro, infine, è il `PATH` dell'immagine "pulita", ovvero l'immagine com'era prima di essere stata modificata aggiungendo del rumore. È necessario e molto utile per le simulazioni, può tranquillamente essere omesso nel caso non si abbia a disposizione l'immagine originale.

Per facilitare l'esecuzione del programma viene fornito un piccolo script in bash, `ATVD` (vedi Fig. 2.6). Grazie alla sua struttura, è possibile richiamare il programma semplicemente scrivendo:

```
./ATVD SIM IMAGE_NAME ORIGINAL_IMAGE_NAME TAG
```

nel caso in cui si voglia fare una simulazione (quindi si ha a disposizione

```
#!/bin/bash

OUT=ATVD.x
DF=FITS
DR=REPORT

case $1 in

        SIM)

                if [ -n "$4" ]; then

                        ./$OUT ./$DF/$2.fits $2 $4 ./$DF/$3.fits > ./$DR/$2_$4.Report

                else

                        ./$OUT ./$DF/$2.fits $2 ignore ./$DF/$3.fits > ./$DR/$2.Report

                fi

                ;;

        DEC)

                if [ -n "$3" ]; then

                        ./$OUT ./$DF/$2.fits $2 $3 > ./$DR/$2_$3.Report

                else

                        ./$OUT ./$DF/$2.fits $2 ignore > ./$DR/$2.Report

                fi

                ;;

        *)

                echo Errore nei parametri.

                ;;
esac
```

Figura 2.6: Codice BASH per ATVD

anche l'immagine originale, non affetta da rumore),
`./ATVD DEC IMAGE_NAME TAG`
nel caso in cui si voglia effettuare una decomposizione senza avere a disposizione l'immagine originale. Il parametro `TAG` è del tutto opzionale: se omesso, verrà ignorato dal programma.
Se il programma viene lanciato in modalità simulazione (con il comando `./ATVD SIM`) verranno eseguiti quattro test statistici di somiglianza tra l'immagine originale (priva di disturbi) e l'output di `ATVD` (la parte struttura). Il primo valore calcolato è lo scarto quadratico medio tra le due immagini (RMSE), definito da

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I(i,j) - S(i,j)\right)^2}{N \cdot M}},$$

dove I rappresenta la matrice associata all'immagine senza rumore e S la matrice associata all'immagine prodotta (entrambe di dimensioni $N \cdot M$). Il coefficiente di variazione (CV) misura invece il rapporto tra il RMSE calcolato sulle stesse immagini e il valore quadratico medio dei pixel della matrice associata all'immagine originale; in formule abbiamo

$$CV = \frac{RMSE}{\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(I(i,j))^2}{N \cdot M}}}.$$

Il PNSR (*Peak-signal to noise ratio*) si esprime come il rapporto, espresso in decibel, tra il valore massimo dell'immagine originale e il RMSE:

$$PNSR = 20 \cdot \log_{10}\left(\frac{\max_{i,j} I(i,j)}{RMSE}\right).$$

Tutti questi valori esprimono in un certo senso una misura di "somiglianza" tra l'immagine originale e l'immagine prodotta dal programma. Nell'articolo [22] gli autori, dopo aver osservato che il semplice calcolo del RMSE e il giudizio dell'occhio umano non sono sufficienti a stabilire con accuratezza se un'immagine "somiglia" sufficientemente ad un'altra, hanno proposto un ulteriore metodo statistico per stabilire con una certa precisione se due immagini possono essere considerate "somiglianti": lo Structure Similarity Index (SSIM). Date due immagini (o porzioni di immagini) rappresentati dalle

matrici $x, y \in X$, lo SSIM è definito da:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

dove μ rappresenta il valore medio (associato alla sua immagine), σ_{xy} la covarianza tra le due matrici e σ^2 la varianza. Per stabilizzare la divisione per piccoli denominatori sono introdotte le costanti $c_1 = (L \cdot 0.01)^2$ e $c_2 = (L \cdot 0.03)^2$ con $L = 2 \cdot \#\text{bitperpixel} - 1$ (range dinamico delle immagini). Lo SSIM varia tra -1 e 1, col valore massimo ottenibile solo da due immagini identiche. Seguendo quanto suggerito da Wang nel suo lavoro [22] il calcolo dello SSIM globale viene effettuato come valore medio degli SSIM calcolati su finestre di 8×8 pixel.

2.7.3 Il file `.REPORT`

Dopo ogni esecuzione del programma tramite lo script `ATVD` viene sempre generato un file `.REPORT` che viene automaticamente salvato in un'apposita cartella preesistente (per default è la cartella `REPORT`). Il file è salvato con il nome `ImageName_Tag.REPORT`. Aprendolo, possiamo osservare:

- Fig. 2.7(a): un riassunto dei parametri scelti, il numero di threads coinvolti nell'esecuzione, le opzioni di output e le dimensioni dell'immagine (espresse in pixel).

- Fig. 2.7(b): il tempo impiegato dall'algoritmo di ricerca per λ e i suoi risultati (voce assente se disattivato), il tempo impiegato dall'algoritmo scelto (solitamente maggiore poiché nel calcolare la soluzione si richiede una tolleranza molto più piccola per il criterio di arresto) e il tempo impiegato dalla scrittura dei file. Per una stima dei tempi impiegati dal programma esterno `R` (con il quale vengono creati tutti i grafici di output) consultare il file `Statistica.Rout` (generato dal programma nella cartella di output).

- Fig. 2.7(c): nel solo caso di modalità simulazione, i risultati dei test di somiglianza (RMSD, PSNR, CV e SSIM) introdotti nel paragrafo precedente.

```
Astro Total Variation Denoiser

Image (Noised):  ./FITS/noise256.fits
Original Image (Clean):  ./FITS/clean256.fits
tau= 0.12
eps= 0.01
eps2= 1e-05
numero di threads= 4
alpha= 1
metodo= TVL1
statistica= SI
autolambda= SI
Salva in= ./PROVE
Nome= CON_PARAMETRI
ni= 256
nj= 256
```

(a)

```
Tempo impiegato per correlazione
Tempo -----------------> 0.869141
imin=  12
Lambda= 0.28243
Tempo impiegato per la soluzione
Tempo -----------------> 1.775391
Tempo impiegato per la scrittura
Tempo -----------------> 0.390000
```

(b)

```
Root-mean-square deviation
RMSD -----------------> 0.022895
Peak-Signal to noise ratio
PSNR -----------------> 33.479109
Coefficiente di variazione
CV -----------------> 0.978539
Structure Similarity Index
SSIM -----------------> 0.963230
```

(c)

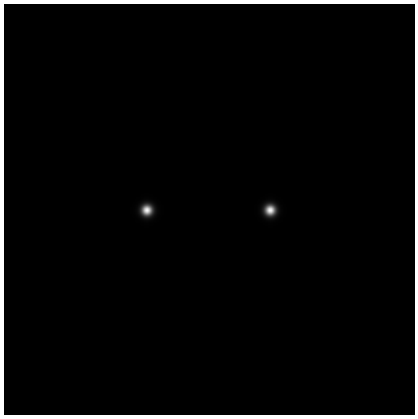
Figura 2.7: Esempio di file `.REPORT`.

2.7.4 Un esempio di applicazione

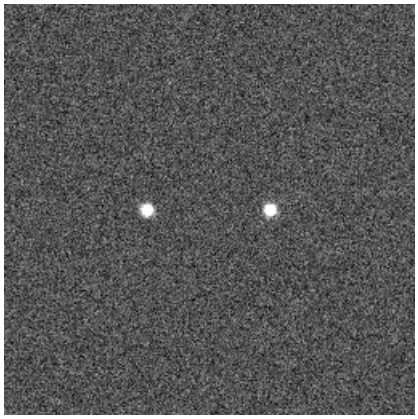
La Fig. 2.8 mostra una semplice simulazione di immagine astronomica contenente due sorgenti (stelle) con l'aggiunta di rumore gaussiano. Dopo aver opportunatamente editato il foglio dei parametri (per questa simulazione abbiamo scelto l'algoritmo A^2BC relativo al problema $TV - G$, la ricerca automatica del miglior parametro λ e un criterio di arresto dell'ordine di 10^{-5}) possiamo lanciare il programma con il seguente comando:

```
./ATVD SIM noise256 clean256
```

L'output del programma viene salvato nella cartella generata appositamente (vedi Fig. 2.10). I risultati prodotti dal programma sono molto buoni: nella Fig. 2.9 possiamo osservare la decomposizione dell'immagine 3.1(b); la prima immagine rappresenta la parte struttura, la seconda la parte texture.



(a) `clean256.fits`

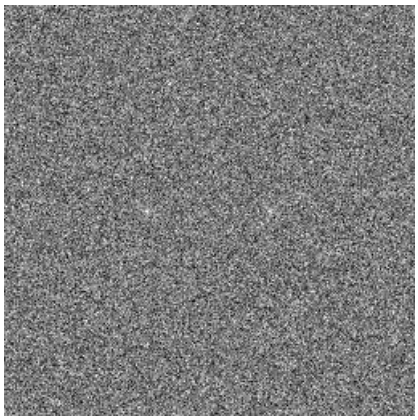


(b) `noise256.fits`

Figura 2.8: Simulazione di una semplice immagine astronomica prima e dopo l'aggiunta di rumore gaussiano.

Nell'ottica del denoising, la prima immagine rappresenta l'immagine "pulita", la seconda il rumore rimosso.

(a) `Structural.fits`



(b) `Texture.fits`

Figura 2.9: Decomposizione dell'immagine 3.1(b) in parte struttura e parte texture

Analisi dell'output

Il file `.Report` associato a questa decomposizione riporta diverse informazioni. In Fig. 2.12 possiamo vedere il tempo di esecuzione del programma,

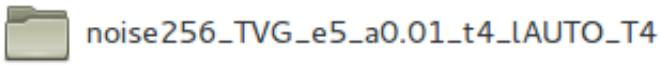


Figura 2.10: Cartella prodotta dal programma tramite l'opzione di output `CON_PARAMETRI`

registrato separatamente durante la ricerca del minimo con l'algoritmo di correlazione, durante l'esecuzione del programma con l'algoritmo scelto (in questo caso A^2BC, il più lento dei tre) e durante la scrittura dei file. Ovviamente tale conteggio non tiene conto del tempo impiegato da `R`; tale valore può essere letto nel file `Statistica.Rout`. Questi tempi sono calcolati su un'esecuzione del programma che sfrutta 4 processori in parallelo. La Fig. 2.11 mostra l'andamento dei tempi di esecuzione al variare del numero dei threads coinvolti nella parallelizzazione.

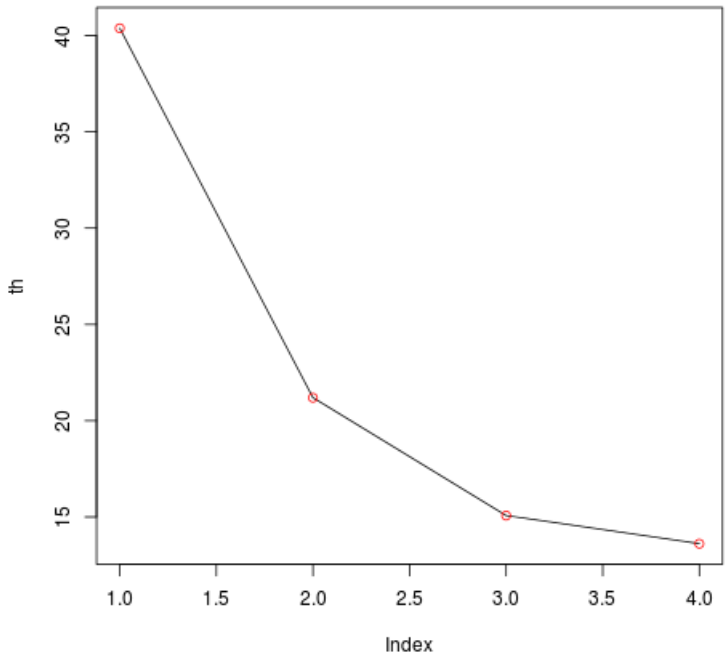


Figura 2.11: Tempi di esecuzione al variare del numero dei threads

In Fig. 2.13 sono riportati invece i valori dei quattro "indicatori di somiglianza" scelti per valutare l'affidabilità della decomposizione. Questi

```
Tempo impiegato per correlazione
Tempo -----------------> 3.021973
Tempo impiegato per la soluzione
Tempo -----------------> 13.613525
Tempo impiegato per la scrittura
Tempo -----------------> 0.380000
```

Figura 2.12: Il tempo impiegato

```
Root-mean-square deviation
RMSE ------------------> 0.002796
Peak-Signal to noise ratio
PSNR ------------------> 51.742302
Coefficiente di variazione
CV ------------------> 0.119514
Structure Similarity Index
SSIM ------------------> 0.991419
```

Figura 2.13: I risultati dei quattro test di somiglianza

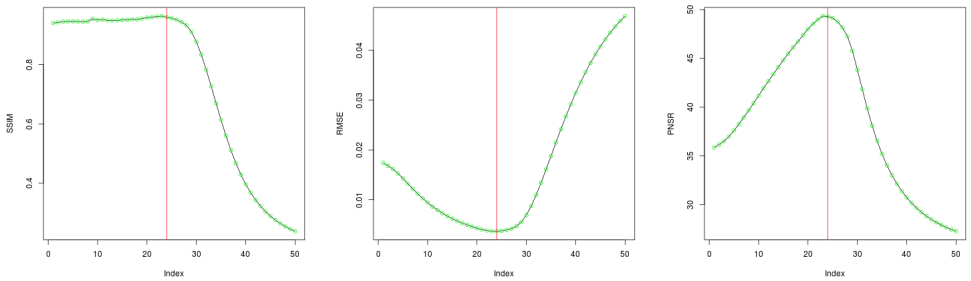
indicatori sono stati descritti nella Sez. 2.7. Ricordandoci che:

- Il diminuire del valore del RMSE (*Root mean square error*) è indicativo di una maggiore somiglianza tra l'immagine senza rumore e quella in output e dunque di una migliore qualità della decomposizione. Stesso discorso vale per il CV (*coefficiente di variazione*, data la loro dipendenza. Il coefficiente di variazione ha il vantaggio di essere indipendente dalla media dei valori dell'immagine analizzata: più è vicino allo 0 e più le due immagini si "somigliano".

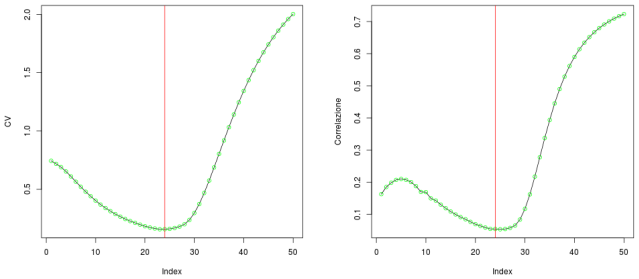
- Il PNSR (*Peak-signal to noise ratio*) e il SSIM (*Structure similarity index*) seguono invece regole inverse; il primo ha una dipendenza inversa con il RMSE, di conseguenza più è alto e meglio è, il secondo è invece limitato tra -1 e 1 per costruzione, dove l'estremo destro rappresenta il punto di massima somiglianza per questo metodo.

Osservando la Fig. 2.13 ci rendiamo subito conto che le cose sono andate abbastanza bene. Ricordiamoci che il parametro λ utilizzato per questa decomposizione è stato calcolato dall'algoritmo di correlazione. Il successivo passo di questa analisi è quello di verificare se tale scelta è stata effettivamente la migliore. Data l'impossibilità di controllare tutte le immagini prodotte dal programma al variare di λ, ci limiteremo ad osservare i grafici dei quattro test di somiglianza scelti effettuati per ogni λ presente nel vettore dei parametri

test (gli stessi che abbiamo sfruttato per calcolare la correlazione tra le due parti della decomposizione ad ogni iterazione dell'algorimo di correlazione). La Fig. 2.22 riporta gli andamenti della correlazione e dei quattro test al variare di λ. Possiamo osservare che il parametro scelto dall'algoritmo di correlazione sembra effettivamente produrre la decomposizione migliore (o almeno quella che contiene la parte struttura più somigliante all'immagine senza disturbo).



(a) SSIM (b) RMSE (c) PNSR



(d) CV (e) Correlazione

Figura 2.14: Andamento della correlazione e dei quattro test di somiglianza al variare di λ. La linea rossa in verticale rappresenta il valore di lambda scelto dall'algoritmo di correlazione. I test sono stati effettuati partendo da un valore iniziale di $\lambda = 1$, diminuendolo ad ogni iterazione moltiplicandolo per 0.9. In questo particolare caso, il valore minimo della correlazione si ottiene per $\lambda = 0.0797664$

Analisi statistica del rumore rimosso

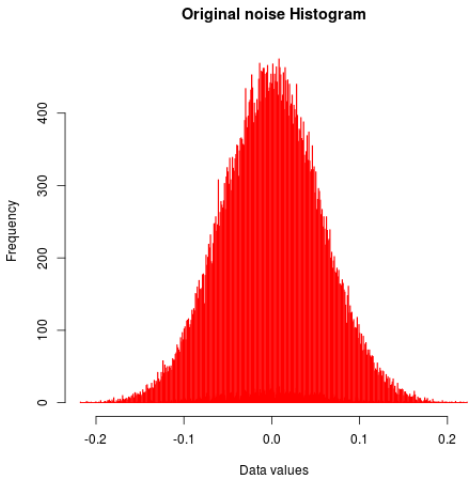
Nell'ottica del denoising, la parte texture dell'immagine rappresenta il rumore rimosso. In primo luogo vogliamo quindi analizzare tale rumore per capire se ci troviamo effettivamente davanti a del rumore gaussiano (ovvero ad una rappresentazione di un vettore aleatorio formato da componenti gaussiane indipendenti). Per compiere tale analisi sfruttiamo il software statistico `R`. Il foglio di comando `Statistica.R` generato da `ATVD` è visibile in Fig. 2.18. Tale file viene poi passato ad `R` in modalità `BATCH`, in modo da generare tutto il suo output come parte dell'esecuzione di `ATVD`. In particolare ci interessano:

- L'istogramma dei valori dei pixel contenuti nella parte texture dell'immagine e l'istogramma dei valori del rumore originale (vedi Fig. 2.15). Questi ultimi possono essere facilmente estratti per sottrazione, avendo a disposizione l'immagine originale priva di disturbo. Nella realtà, ci dovremmo accontentare di un singolo istogramma. Osservandoli ci rendiamo conto che entrambi i dati sembrano essere estratti da fonti gaussiane; inoltre, sovrapponendoli, ci possiamo rendere conto di quanto rumore è effettivamente stato rimosso.

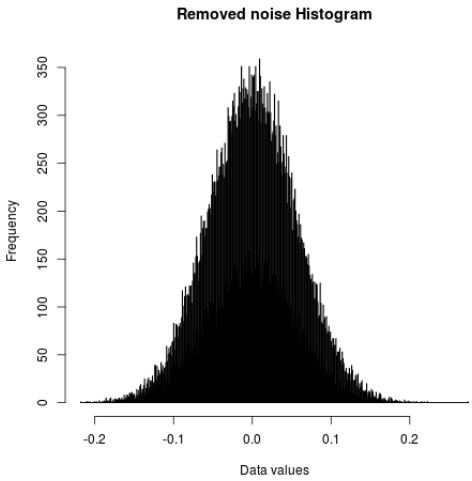
- La rappresentazione della funzione di autocorrelazione delle serie differenziate relative al rumore rimosso e al rumore originale con diversi "lag" (Fig. 2.16). Dalla teoria sappiamo che il rumore gaussiano scorrelato presenta un valore di autocorrelazione pari a $-\frac{1}{2}$ quando il lag è uguale a 1, 0 quando il lag è maggiore. Dai grafici possiamo vedere come tale condizione è sufficientemente rispettata sia dal rumore originale che dal rumore rimosso.

- La comparazione tra quantili della distribuzione seguita dalla serie da analizzare e della distribuzione normale (Q-Q *norm*) applicata sia al rumore originale che al rumore rimosso (Fig. 2.17). In questo caso possiamo osservare come entrambe le serie analizzate sembrino seguire una distribuzione normale.

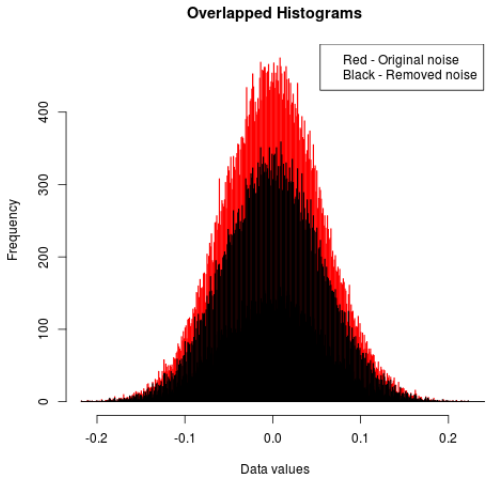
.



(a) Istogramma rumore originale

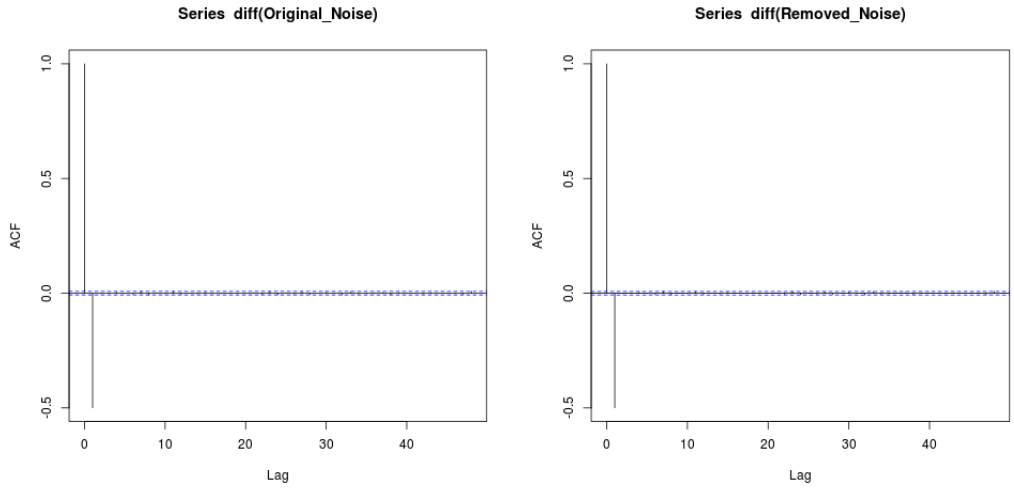


(b) Istogramma rumore rimosso



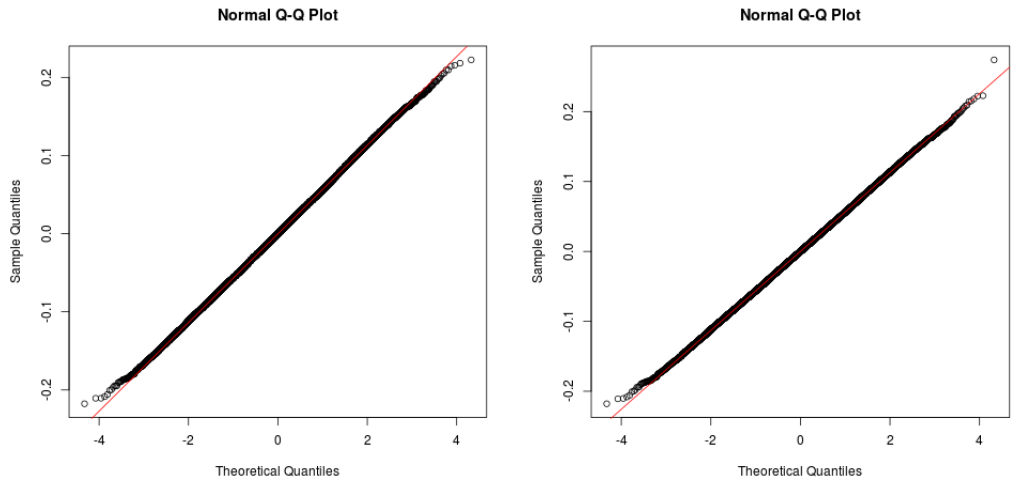
(c) Istogrammi sovrapposti

Figura 2.15:



(a) Autocorrelation function (Acf) del rumore orginale differenziato (b) Acf del rumore rimosso differenziato

Figura 2.16:



(a) *Q-Q norm* relativo al rumore originale (b) *Q-Q norm* relativo al rumore rimosso

Figura 2.17:

```
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/Istogramma_rumore_rimosso.png")
Removed_Noise=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/datatexture.txt")
hist(Removed_Noise,xlim = range(Removed_Noise),nclass=1000,border= 1, col= 1, main="Removed noise
     Histogram", xlab="Data values")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/Istogramma_rumore_originale.png")
Original_Noise=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/datadiff.txt")
hist(Original_Noise,xlim = range(Original_Noise),nclass=1000,border= 2, col= 2, main="Original noise
     Histogram", xlab="Data values")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/Istogrammi_sovrapposti.png")
hist(Original_Noise,xlim = range(Original_Noise),nclass=1000,border= 2, col= 2, main="Overlapped
     Histograms", xlab="Data values")
hist(Removed_Noise,xlim = range(Removed_Noise),nclass=1000, border= 1, add=T, col= 1, main="
     Overlapped Histograms", xlab="Data values")
temp <- legend("topright", legend=c("Red - Original noise", "Black - Removed noise"))
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/QQplot_rumore_originale.png")
qqnorm(Original_Noise);qqline(Original_Noise, col = 2)
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/acf_originale.png")
acf(Original_Noise)
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/acf_diff_originale.png")
acf(diff(Original_Noise))
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/Correlazione.png")
Correlazione=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/datacorr.txt")
plot(Correlazione,type = "l")
points(Correlazione, col="green")
abline(v=24,col = "red")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/RMSE.png")
RMSE=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/RMSE.txt")
plot(RMSE,type = "l")
points(RMSE, col="green")
abline(v=24,col = "red")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/PNSR.png")
PNSR=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/PNSR.txt")
plot(PNSR,type = "l")
points(PNSR, col="green")
abline(v=24,col = "red")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/CV.png")
CV=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/CV.txt")
plot(CV,type = "l")
points(CV, col="green")
abline(v=24,col = "red")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/SSIM.png")
SSIM=scan(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/SSIM.txt")
plot(SSIM,type = "l")
points(SSIM, col="green")
abline(v=24,col = "red")
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/QQplot_rumore_rimosso.png")
qqnorm(Removed_Noise);qqline(Removed_Noise, col = 2)
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/acf_rimosso.png")
acf(Removed_Noise)
dev.off()
png(file="./PROVE/noise256_TVG_e5_a0.01_t4_lAUTO_T4/acf_diff_rimosso.png")
acf(diff(Removed_Noise))
dev.off()
```

Figura 2.18: Script in R generato da ATVD

2.7.5 Analisi della scalabilità

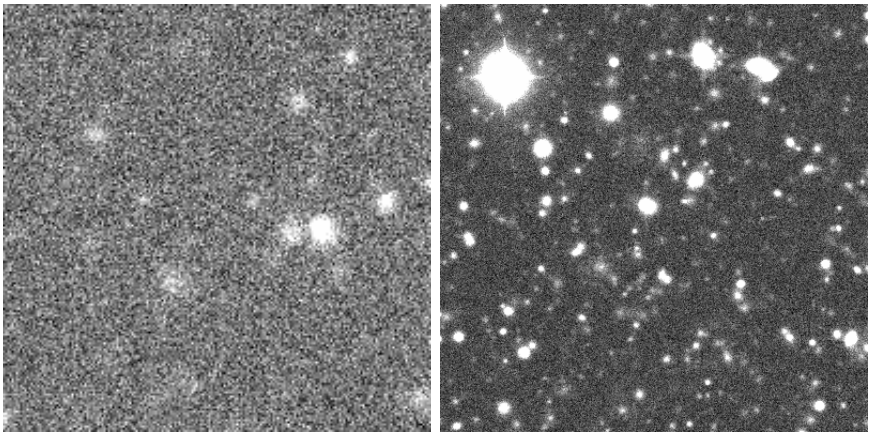
Per migliorare i tempi di esecuzione del programma abbiamo deciso di proporre una parallelizzazione del codice C++ sfruttando `OpenMP`. `OpenMP` (*Open Multiprocessing*) è un API multipiattaforma per la creazione di applicazioni parallele su sistemi a memoria condivisa. È supportata da vari linguaggi di programmazione, tra cui il C++. In questa sottosezione proponiamo un analisi della scalabilità del programma mostrando sia un confronto tra i tempi di calcolo al variare del numero di threads coinvolti sia misurandone Speed-Up ed Efficienza. Ricordiamo che lo Speed-Up di un programma lanciato su n threads è definito come il rapporto tra il tempo di esecuzione seriale (1 thread) del programma e il suo tempo di esecuzione, mentre l'Efficienza rapporta lo speed-up col valore lineare. In formule abbiamo:

$$S(n) = \frac{T(1)}{T(n)},$$

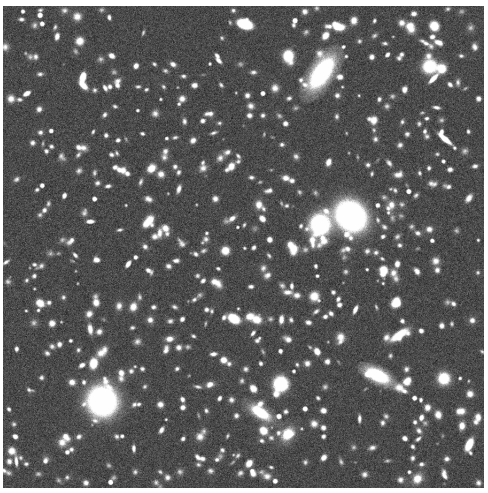
$$E(n) = \frac{S(n)}{n}.$$

Per i test abbiamo scelto tre immagini astronomiche di differenti dimensioni: la prima è una 200×200 pixels, la seconda una 400×400 mentre la terza è una 900×900. Le immagini scelte sono riportate in Fig.2.19. Nelle Fig. REF sono riportati i grafici relativi ai tempi di esecuzione dell'algoritmo di correlazione e del programma al variare del metodo scelto per la decomposizione in funzione del numero di threads coinvolti nell'esecuzione del programma.

In Fig. 2.23 sono riportati i risultati relativi al calcolo dello Speed-Up e dell'Efficienza sulle tre immagini al variare del numero di threads coinvolti.

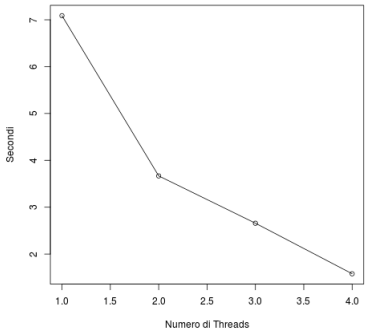


(a) 200×200 pixels (b) 400×400 pixels

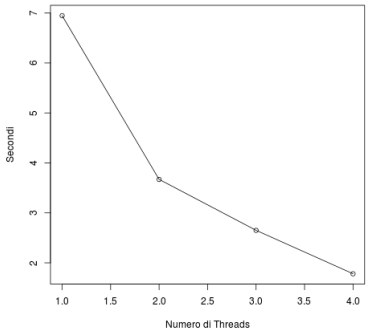


(c) 900×900 pixels

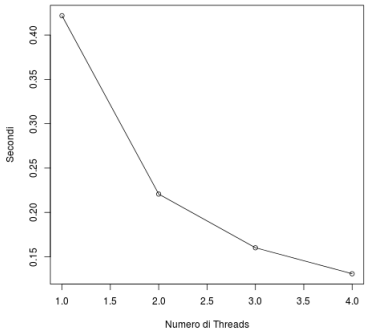
Figura 2.19: Immagini test per misura di scalabilità di `ATVD`



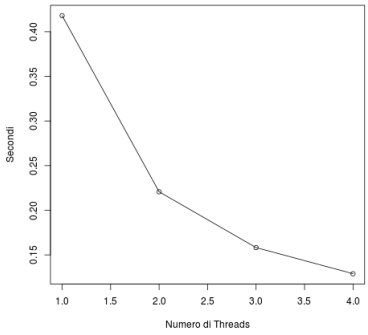
(a) Correlazione $TV - L^1$



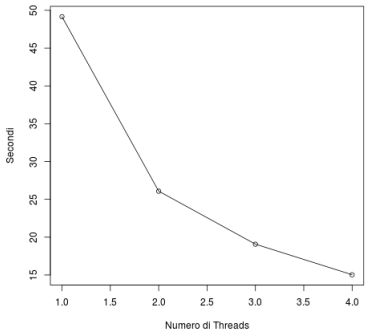
(b) Correlazione $TV - L^2$ e $TV - G$



(c) $TV - L^1$

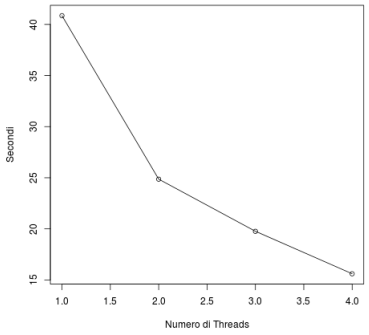


(d) $TV - L^2$

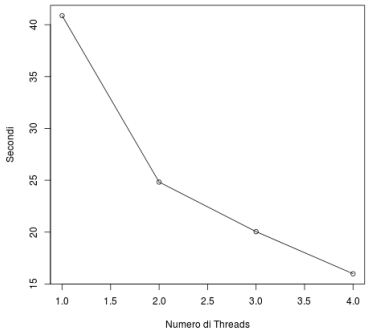


(e) $TV - G$

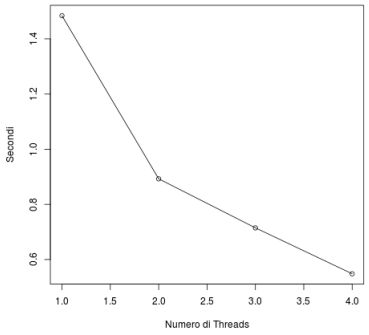
Figura 2.20: Grafico dei tempi di esecuzione al variare del numero di threads coinvolti per 2.19(a)



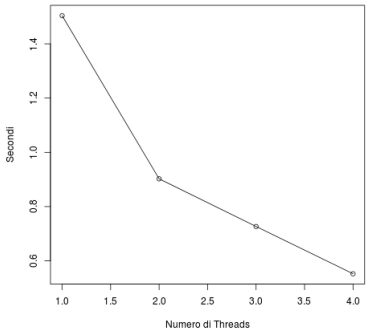
(a) Correlazione $TV - L^1$



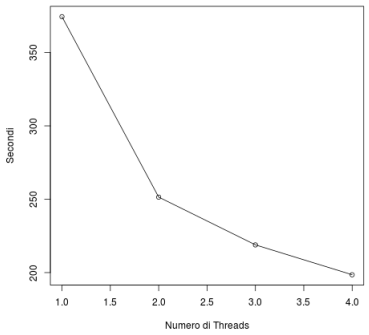
(b) Correlazione $TV - L^2$ e $TV - G$



(c) $TV - L^1$

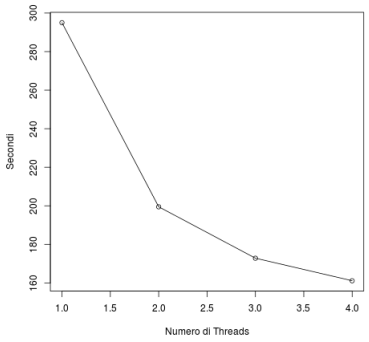


(d) $TV - L^2$

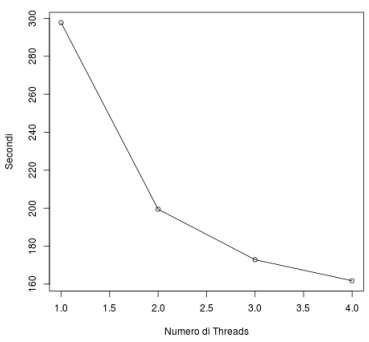


(e) $TV - G$

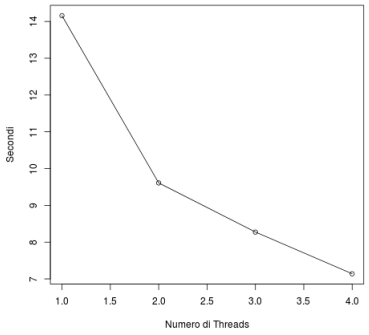
Figura 2.21: Grafico dei tempi di esecuzione al variare del numero di threads coinvolti per 2.19(b)



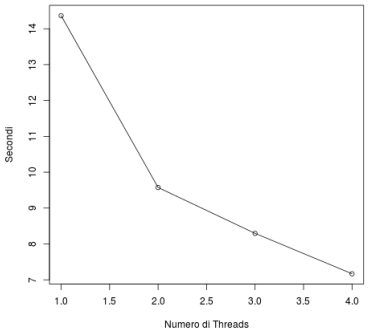
(a) Correlazione $TV - L^1$



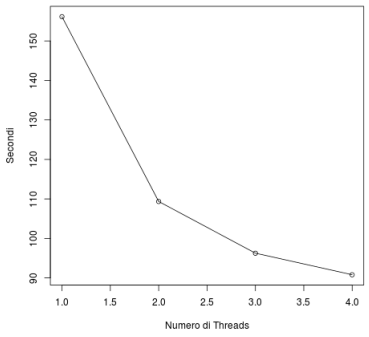
(b) Correlazione $TV - L^2$ e $TV - G$



(c) $TV - L^1$



(d) $TV - L^2$



(e) $TV - G$

Figura 2.22: Grafico dei tempi di esecuzione al variare del numero di threads coinvolti per 2.19(c)

```
Dimensioni immagine: 900x900 pixels
        S[1]     S[2]     S[3]     S[4]         E[1]      E[2]      E[3]      E[4]
TVL2    1.000000 1.500408 1.731811 2.003759     1.0000000 0.7502040 0.5772702 0.5009399
TVL1    1.000000 1.473374 1.711284 1.982985     1.0000000 0.7366870 0.5704280 0.4957462
TVG     1.000000 1.428036 1.622165 1.719688     1.0000000 0.7140178 0.5407215 0.4299219
CORL1   1.000000 1.478838 1.706934 1.830865     1.0000000 0.7394190 0.5689780 0.4577162
COR     1.000000 1.492553 1.722140 1.840086     1.0000000 0.7462764 0.5740467 0.4600215

Dimensioni immagine: 400x400 pixels
        S[1]     S[2]     S[3]     S[4]         E[1]      E[2]      E[3]      E[4]
TVL2    1.000000 1.666666 2.069894 2.722771     1.0000000 0.8333330 0.6899645 0.6806926
TVL1    1.000000 1.663020 2.076502 2.706550     1.0000000 0.8315100 0.6921673 0.6766376
TVG     1.000000 1.489137 1.710566 1.885907     1.0000000 0.7445685 0.5701887 0.4714768
CORL1   1.000000 1.643284 2.067299 2.617493     1.0000000 0.8216418 0.6890997 0.6543731
COR     1.000000 1.646291 2.038776 2.556316     1.0000000 0.8231453 0.6795921 0.6390789

Dimensioni immagine: 200x200 pixels
        S[1]     S[2]     S[3]     S[4]         E[1]      E[2]      E[3]      E[4]
TVL2    1.000000 1.893808 2.641979 3.242432     1.0000000 0.9469038 0.8806597 0.8106081
TVL1    1.000000 1.911506 2.634150 3.223890     1.0000000 0.9557528 0.8780502 0.8059725
TVG     1.000000 1.885186 2.577939 3.271085     1.0000000 0.9425929 0.8593131 0.8177713
CORL1   1.000000 1.932375 2.666422 3.895113     1.0000000 0.9661874 0.8888073 0.9737783
COR     1.000000 1.893028 2.621223 3.899364     1.0000000 0.9465141 0.8737410 0.9748409
```

Figura 2.23: Calcolo dello Speed-Up e dell'Efficienza sui risultati relativi alle tre immagini test

Capitolo 3

Simulazioni numeriche

In questo capitolo proporremo una serie di risultati ottenuti decomponendo varie immagini con diversi livelli di rumore additivo. Inizieremo dal semplice denoising di un'immagine fotografica con aggiunta di rumore gaussiano, poi passeremo a decomporre immagini geometriche appositamente costruite per studiare gli effetti della decomposizione sui valori dei singoli pixel. Infine osserveremo il comportamento di `ATVD` applicandolo ad una simulazione di un'immagine astronomica più complessa rispetto a quella già proposta nel precedente capitolo.

3.1 Lena: Immagine fotografica con aggiunta di rumore gaussiano

La prima immagine fotografica scelta per testare le capacità degli algoritmi di decomposizione usati è la fotografia di Lena. Essa è stata modificata aggiungendo del rumore gaussiano a media nulla e varianza di 5×10^{-2} tramite `MATLAB`. In fondo alla sezione sono riportati i report relativi alle tre prove effettuate (dove tra le altre cose sono riportati i parametri scelti per i test).

Per prima cosa, osserviamo l'andamento della correlazione calcolata tra la parte struttura e la parte texture dell'immagine decomposta al variare di λ. Per questo test abbiamo scelto un vettore di λ di 100 elementi ordinati in

(a) `Original.png` (b) `Noise.png`

Figura 3.1: Immagine fotografica con e senza aggiunta di rumore gaussiano.

senso decrescente, tali che:

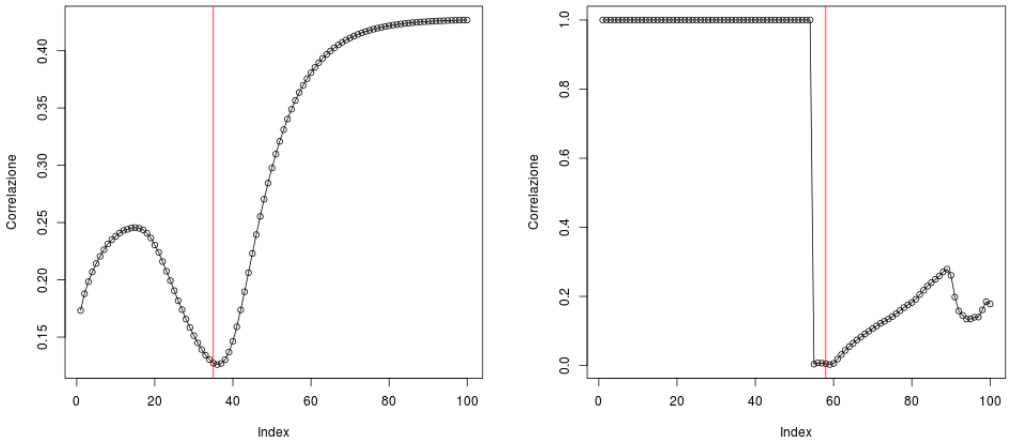
$$\lambda_0 = 1000, \qquad \lambda_{n+1} = 0.9 \cdot \lambda_n, \qquad n \in \{0, \ldots, 99\}.$$

I risultati dell'algoritmo di correlazione sono mostrati in Fig. 3.2. Anche qui, come nella precedente simulazione, possiamo notare che i quattro test scelti per l'analisi di somiglianza tra l'immagine "ripulita" e l'immagine originale (descritti nella sezione 2.7) confermano la scelta automatica del parametro λ (vedi Fig. 3.3).

Dopo aver selezionato il corretto parametro il programma passa quindi alla decomposizione vera e propria. I risultati sono riportati in Fig. 3.4

Analisi del rumore

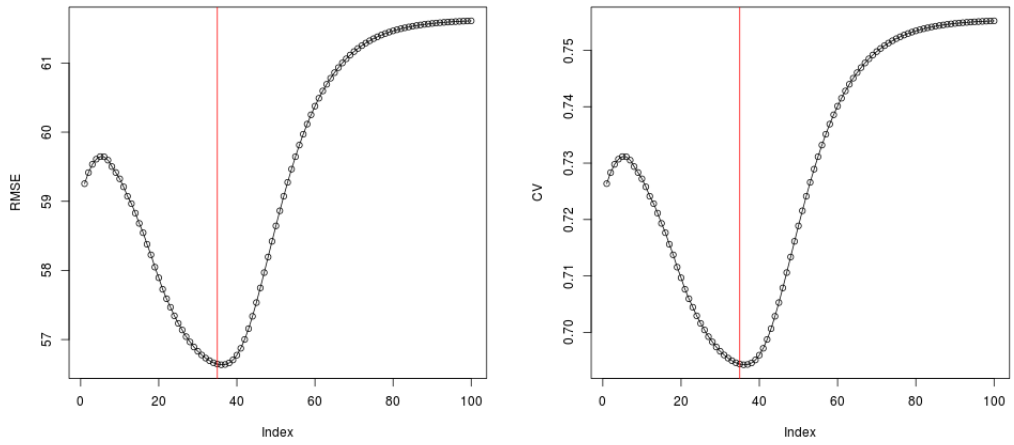
Come abbiamo già detto all'inizio della sezione, l'immagine è stata sporcata con del rumore gaussiano a media nulla e varianza dell'ordine di 10^{-2}. Tramite l'utilizzo del software statistico `R`, possiamo ottenere una prima analisi grafica del rumore rimosso costruendo gli istogrammi relativi ai residui prodotti da ogni singolo algoritmo e confrontandoli con il residuo originale, ovvero il rumore aggiunto. Per "residuo" di un algoritmo intendiamo la



(a) `Correlazione_TVL2.png` (b) `Correlazione_TVL1.png`

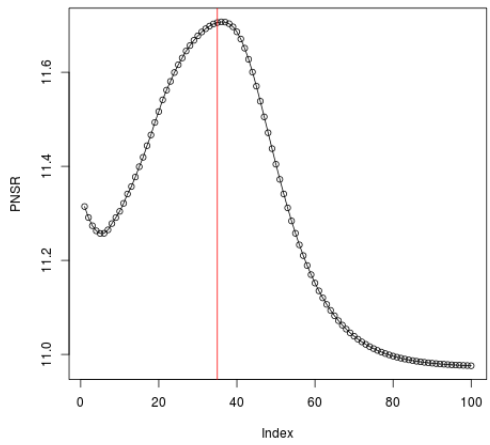
Figura 3.2: Grafici delle correlazioni tra parte struttura e parte texture calcolati per i metodi $TV - L^2$ e $TV - G$ (3.13(a)) e per il metodo $TV - L^1$ (3.2(b)). La retta verticale in rosso rappresenta l'indice del valore minimo.

differenza pixel per pixel tra il valore originale dell'immagine non disturbata e il valore dell'immagine "ripulita" dal programma. Ovviamente tale analisi è possibile soltanto nell'ambito della simulazione. Se non si dispone dell'immagine originale si può comunque avere un'idea del rumore rimosso analizzando i valori contenuti nella parte texture dell'immagine. Tale analisi però risulta veramente accurata solo nel caso di perfetta somiglianza tra la somma di struttura e texture e l'immagine da decomporre (la condizione $f = u + v$, vedi sezioni 2.5 e 2.4). Osservando gli istogrammi prodotti ci accorgiamo subito che nessuno di essi presenta la classica forma a campana di Gauss. Questo è dovuto al fatto che l'algoritmo di decomposizione riconosce come parte texture dell'immagine anche i bordi delle figure presenti (come è evidente dalla Fig. 3.4). D'altra parte però l'immagine struttura risulta significativamente ripulita dal rumore. Aumentando la sensibilità per il criterio di arresto del programma (quindi diminuendo il parametro `EPS[SOL]`, vedi 2.7.2) è possibile ottenere risultati leggermente più accurati: quelli selezionati sono i risultati migliori dal punto di vista del rapporto qualità/tempo impiegato dal programma.

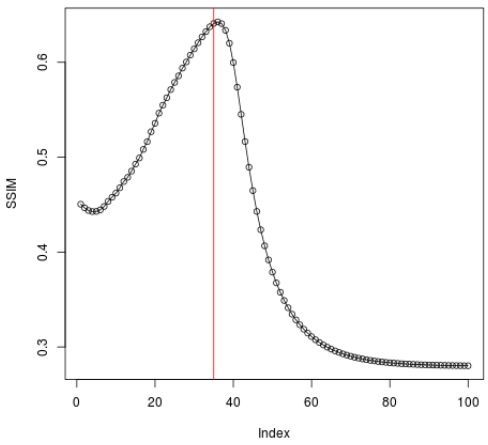


(a) `RMSE.png`

(b) `CV.png`



(c) `PNSR.png`

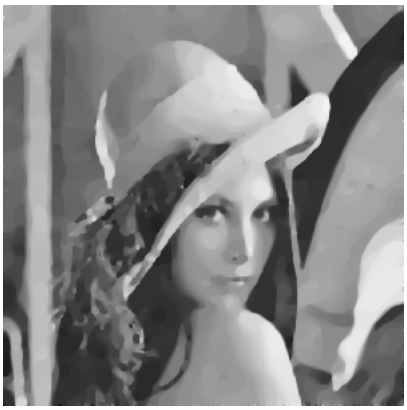


(d) `SSIM.png`

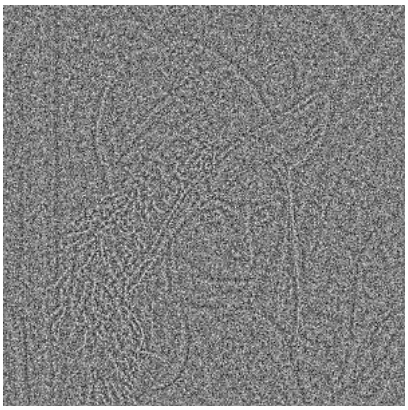
Figura 3.3: Grafici dei risultati dei test di somiglianza al variare del parametro λ. Possiamo notare come la scelta automatica del parametro (evidenziata dalla retta verticale rossa) sia la migliore tra tutte quelle considerate.

(a) $TV - L^2$: Struttura

(b) $TV - L^2$: Texture

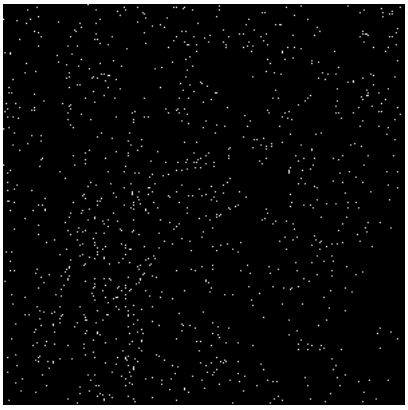


(c) $TV - G$: Struttura



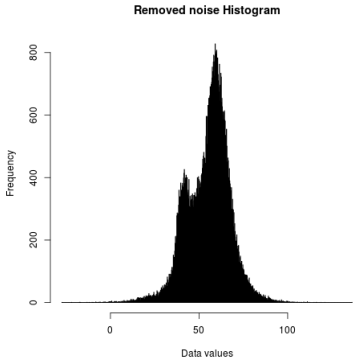
(d) $TV - G$: Texture

(e) $TV - L^1$: Struttura

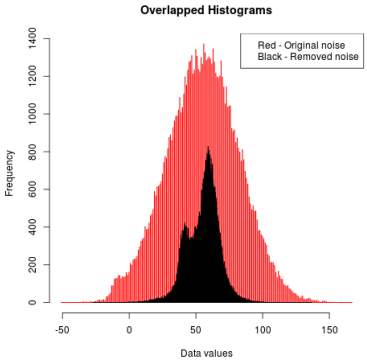


(f) $TV - L^1$: Texture

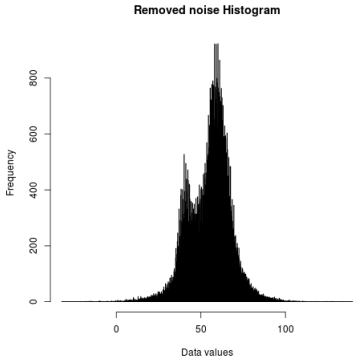
Figura 3.4: Immagini di output del programma per i tre metodi implementati.



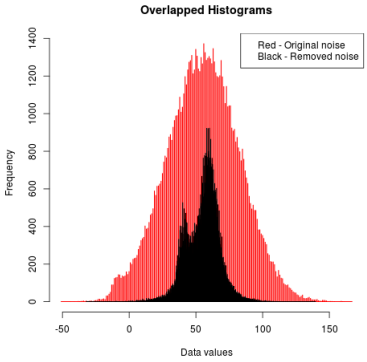
(a) $TV - L^2$



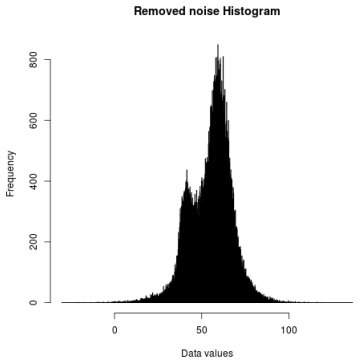
(b)



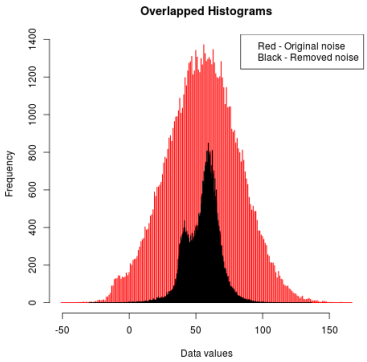
(c) $TV - G$



(d)



(e) $TV - L^1$



(f)

Figura 3.5: Istogrammi del rumore rimosso prodotti dai tre algoritmi implementati e sovrapposizione con istogramma relativo al rumore originale.

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/lenamod.fits            Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/lena.fits       Tempo -----------------> 6.698730
tau= 0.12                                       imin=  35
eps= 0.001                                      Lambda= 25.0316
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 126.443604
alpha= 1                                        Root-mean-square deviation
metodo= TVG                                     RMSE ------------------> 56.722552
statistica= SI                                  Peak-Signal to noise ratio
autolambda= SI                                  PSNR ------------------> 11.694015
Salva in= ./PROVE                               Coefficiente di variazione
Nome= CON_PARAMETRI                             CV ------------------> 0.695318
ni= 300                                         Structure Similarity Index
nj= 300                                         SSIM ------------------> 0.634138
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.100032
```

Figura 3.6: `Report` di Lena con l'algoritmo A^2BC $(TV - G)$

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/lenamod.fits            Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/lena.fits       Tempo -----------------> 6.781006
tau= 0.12                                       imin=  35
eps= 0.001                                      Lambda= 25.0316
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 0.357666
metodo= TVL2                                    Root-mean-square deviation
statistica= SI                                  RMSE ------------------> 56.635345
autolambda= SI                                  Peak-Signal to noise ratio
Salva in= ./PROVE                               PSNR ------------------> 11.707379
Nome= CON_PARAMETRI                             Coefficiente di variazione
ni= 300                                         CV ------------------> 0.694249
nj= 300                                         Structure Similarity Index
                                                SSIM ------------------> 0.643159
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.110000
```

Figura 3.7: `Report` di Lena con l'algoritmo di Chambolle $(TV - L^2)$

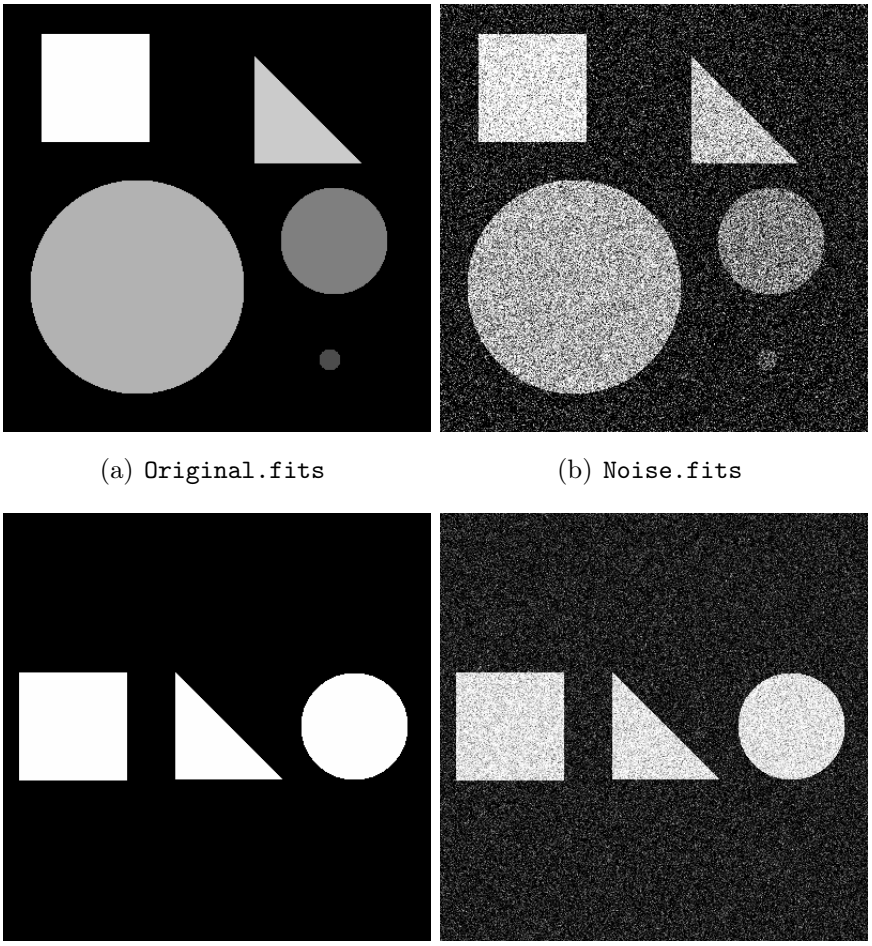
```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/lenamod.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/lena.fits     Tempo -----------------> 143.590576
tau= 0.12                                     imin=  58
eps= 0.001                                    Lambda= 2.21853
eps2= 0.001                                   Tempo impiegato per la soluzione
numero di threads= 4                          Tempo -----------------> 2.959717
alpha= 25                                     Root-mean-square deviation
metodo= TVL1                                  RMSE ------------------> 56.620168
statistica= SI                                Peak-Signal to noise ratio
autolambda= SI                                PSNR ------------------> 11.709707
Salva in= ./PROVE                             Coefficiente di variazione
Nome= CON_PARAMETRI                           CV ------------------> 0.694063
ni= 300                                       Structure Similarity Index
nj= 300                                       SSIM ------------------> 0.641912
                                              Tempo impiegato per la scrittura
                                              Tempo -----------------> 0.110016
```

Figura 3.8: `Report` di Lena con l'algoritmo $Soft-Thresholding$ $(TV-L^1)$

3.2 Immagini geometriche: effetto degli algoritmi sui valori dei pixel

In questa sezione mostreremo dei risultati relativi al lavoro del sofware `ATVD` su due immagini geometriche interamente costruite (e "sporcate") tramite l'utilizzo di `MATLAB`. Lo scopo è quello di analizzare come variano i valori dei pixel dopo la decomposizione. Data la grande quantità di dati effettueremo questa analisi tramite l'utilizzo di una ricostruzione 3-D prima e dopo la decomposizione e, nel caso dell'immagine 3.9(c), anche tramite l'osservazione di un grafico bidimensionale ottenuto intersecando la ricostruzione 3-D dell'immagine con un piano parallelo al piano generato da un versore sull'asse z e un versore sull'asse y e passante per il diametro del cerchio.
La Fig. 3.9 rappresenta le due immagini scelte prima e dopo l'aggiunta di rumore gaussiano. L'immagine 3.9(a) è costruita in modo tale che:

- I pixel appartenenti allo sfondo abbiano valore 0;
- I pixel che costituiscono il quadrato abbiano valore 1;
- I pixel che costituiscono il triangolo abbiano valore 0.8;
- I pixel che costituiscono il cerchio più grande abbiano valore 0.7;



(a) `Original.fits` (b) `Noise.fits`

(c) `Original.fits` (d) `Noise.fits`

Figura 3.9: Le due immagini geometriche costruite con `MATLAB` prima e dopo l'aggiunta di rumore gaussiano.

- I pixel che costituiscono il cerchio "medio" (raggio dimezzato rispetto al più grande) abbiano valore 0.5;

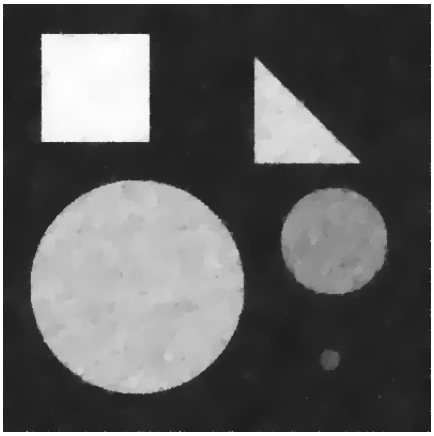
- I pixel che costituiscono il cerchio piccolo (raggio diviso per 10 rispetto al più grande) abbiano valore 0.3.

L'immagine 3.9(c), invece, è costituita da tre figure geometriche con lo stesso valore per ogni pixel (1). Graficamente, tutte queste informazioni sono rappresentate da una mappatura a scala di grigi che associa al valore minimo (0) il colore nero e al valore massimo (1) il colore bianco. Il software di visualizzazione per immagini `.fits` "`SAOImage DS9`" (`http://ds9.si.edu/site/Home.html`) permette di visualizzare immediatamente il valore di ogni singolo pixel semplicemente passandoci sopra con il cursore del mouse.

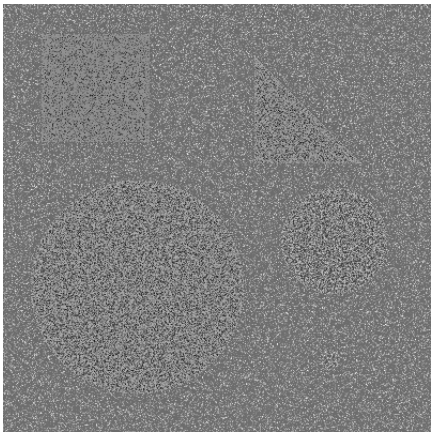
Le figure 3.10 e 3.12 ci mostrano i risultati della decomposizione effettuata con `ATVD` e i relativi grafici riguardanti l'algoritmo automatico di selezione del parametro λ (figure 3.11 e 3.13). Anche in questo caso l'algoritmo sembra aver selezionato il parametro λ con il miglior output possibile, come testimoniano i risultati dei quattro algoritmi di "somiglianza". Possiamo osservare che tutte le immagini prodotte risultano oggettivamente prive di rumore, anche se non proprio perfette. Per osservare meglio quanto accade al livello dei valori dei pixel, sfruttiamo il software `Gnuplot` per fornire una rappresentazione tridimensionale dell'immagine priva di rumore e delle immagini "ripulite" prodotte dal nostro programma (vedi Fig. 3.15 e Fig. 3.18(a)).

Le tabelle mostrate in Fig. 3.16 e 3.17 raccolgono le medie dei valori osservati nei pixel che formano gli oggetti geometrici delle due immagini prima e dopo la decomposizione. Osserviamo come tutti gli algoritmi tendono ad innalzare il livello minimo dei valori dei pixel dell'immagine nei punti dove tali livelli sono più bassi e come al contrario tendono ad abbassarli là dove sono più alti. Nella prima immagine utilizzando i metodi $TV-G$ e $TV-L^1$ addirittura scompare un elemento (ovvero il suo valore viene abbassato allo stesso livello del background rialzato, di fatto scomparendo).

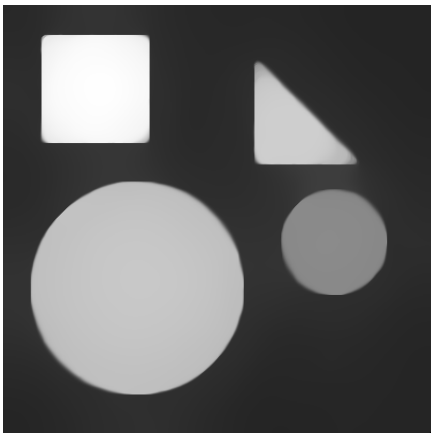
Al fine di renderci conto meglio di quanto questo fenomeno di smussamento dei bordi e di alterazione numerica dei pixel influisca sull'immagine originale, possiamo analizzare la seconda immagine proposta (Fig. 3.9(c)) da un altro punto di vista. Consideriamo le rappresentazioni 3-D delle immagi-



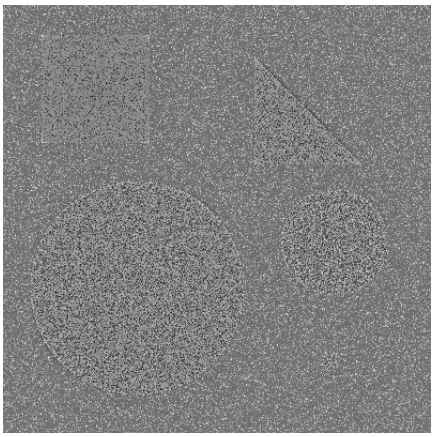
(a) $TV - L^2$: Struttura



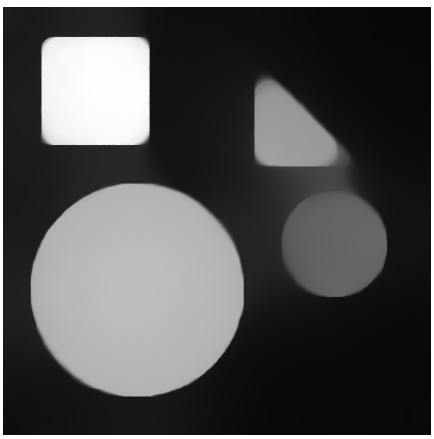
(b) $TV - L^2$: Texture



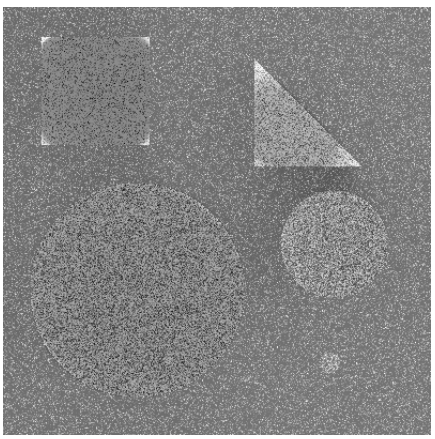
(c) $TV - G$: Struttura



(d) $TV - G$: Texture

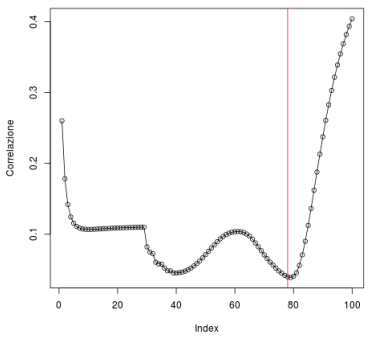


(e) $TV - L^1$: Struttura

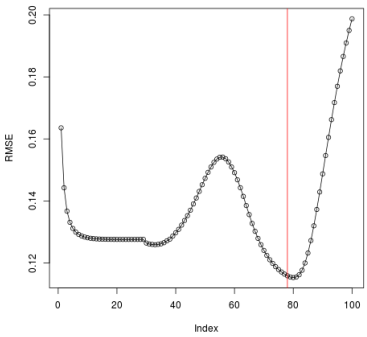


(f) $TV - L^1$: Texture

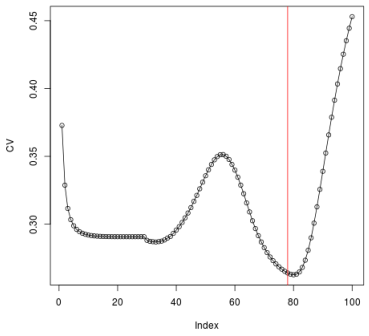
Figura 3.10: Immagini di output del programma per i tre metodi implementati.



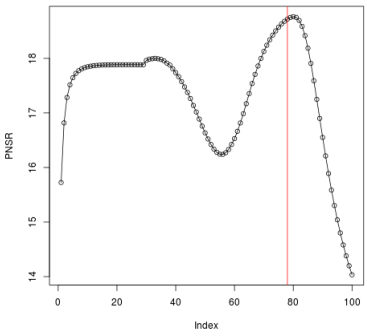
(a) `Correlazione.png`



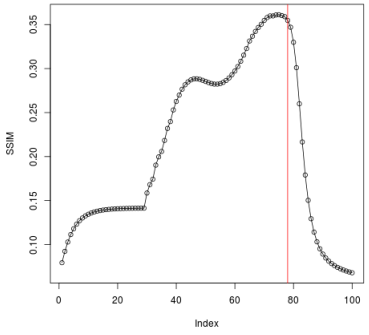
(b) `RMSE.png`



(c) `CV.png`



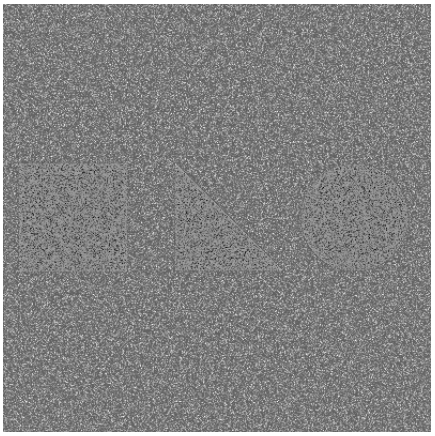
(d) `PNSR.png`



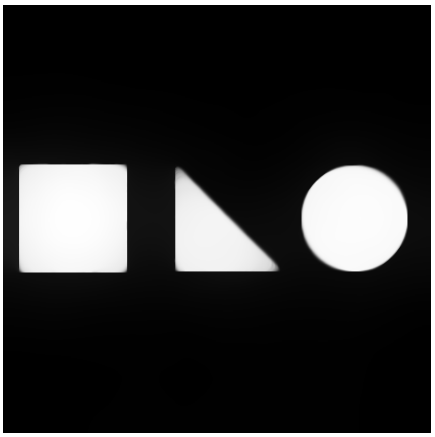
(e) `SSIM.png`

Figura 3.11: Grafici dei risultati dei test di somiglianza al variare del parametro λ.

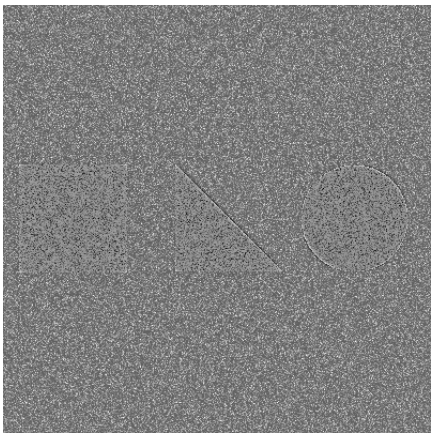
(a) $TV - L^2$: Struttura



(b) $TV - L^2$: Texture

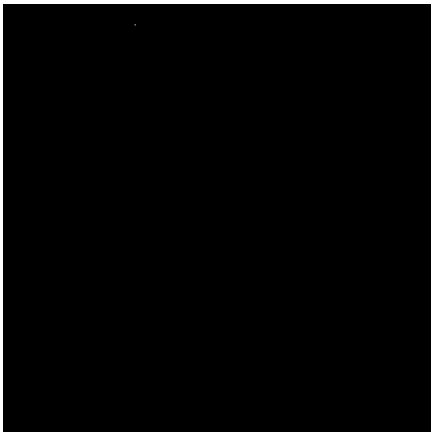


(c) $TV - G$: Struttura



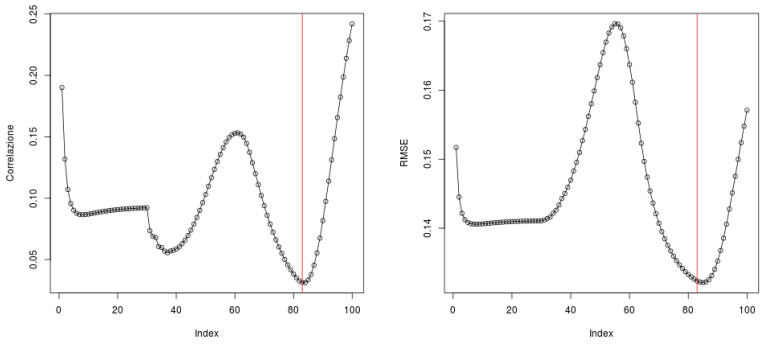
(d) $TV - G$: Texture

(e) $TV - L^1$: Struttura



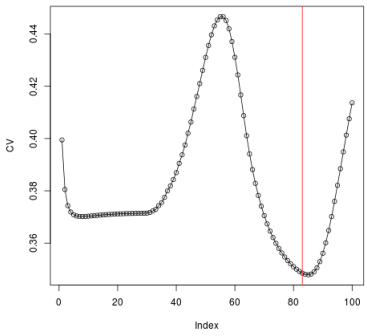
(f) $TV - L^1$: Texture

Figura 3.12: Immagini di output del programma per i tre metodi implementati.

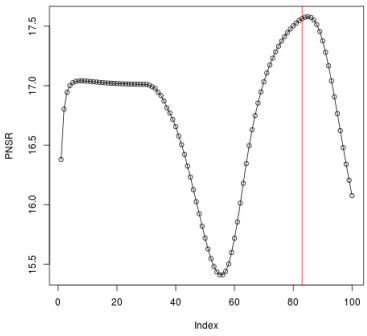


(a) `Correlazione.png`

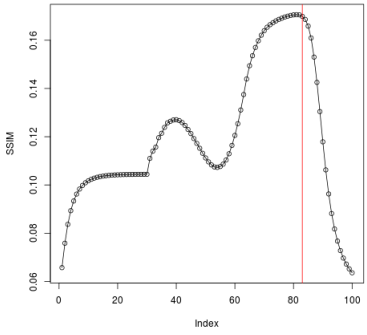
(b) `RMSE.png`



(c) `CV.png`

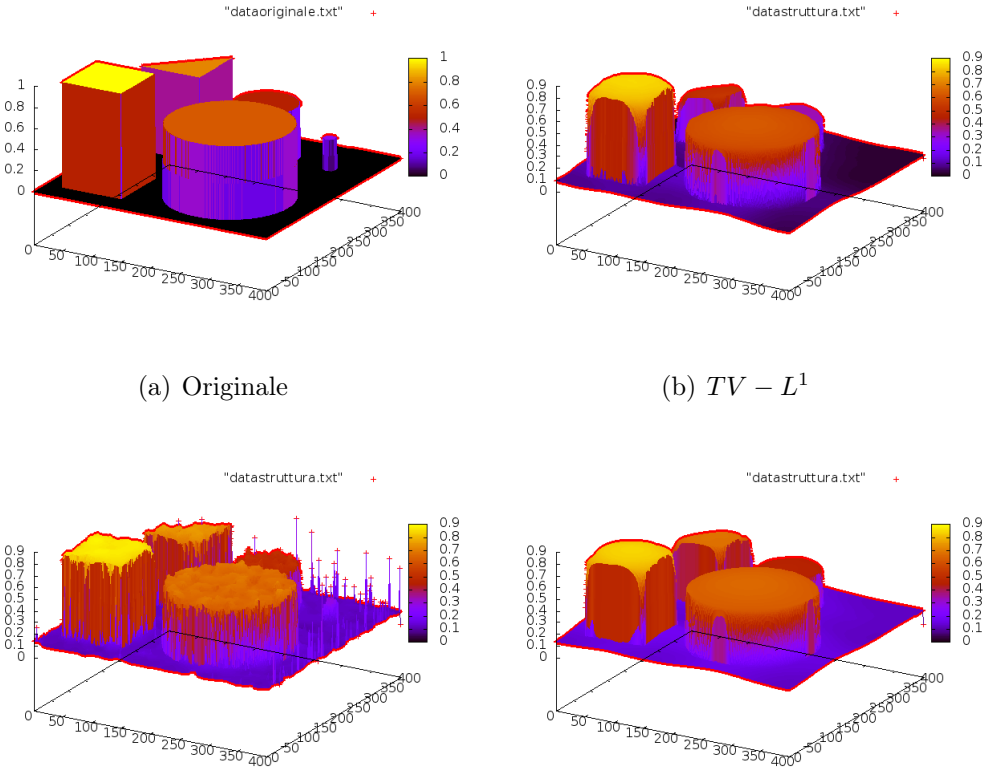


(d) `PNSR.png`



(e) `SSIM.png`

Figura 3.13: Grafici dei risultati dei test di somiglianza al variare del parametro λ.



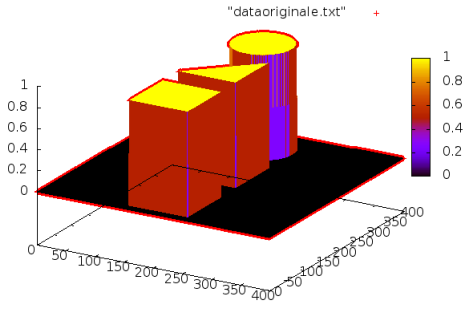
(a) Originale

(b) $TV - L^1$

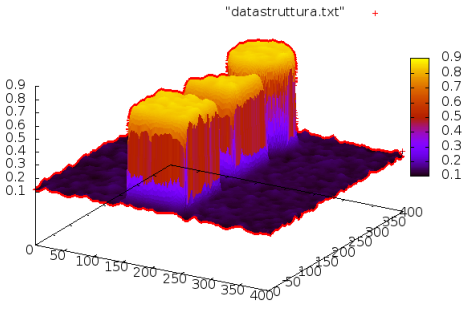
(c) $TV - L^2$

(d) $TV - G$

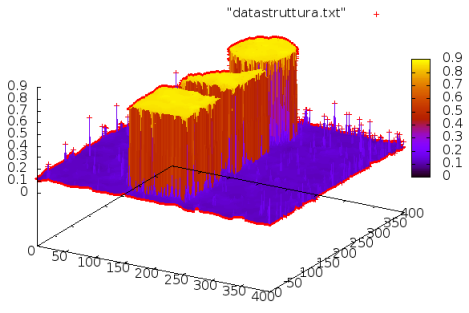
Figura 3.14: Rappresentazione 3d dell'immagine priva di rumore e delle immagini struttura prodotte da `ATVD`.



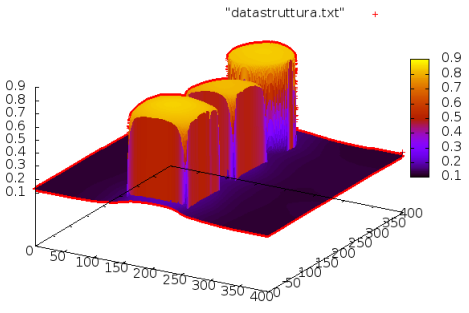
(a) Originale



(b) $TV - L^1$



(c) $TV - L^2$



(d) $TV - G$

Figura 3.15: Rappresentazione 3d dell'immagine priva di rumore e delle immagini struttura prodotte da `ATVD`.

	Background	Quadrato	Triangolo	Cerchio1	Cerchio2	Cerchio3
Originale	0	1	0.8	0.7	0.5	0.3
$TV - L^1$	0.05	0.83	0.51	0.61	0.32	0.04
$TV - L^2$	0.13	0.86	0.72	0.65	0.50	0.30
$TV - G$	0.13	0.84	0.68	0.66	0.46	0.13

Figura 3.16: Tabella delle medie dei valori dei pixel di ogni componente dell'immagine 3.9(a) prima e dopo la decomposizione con `ATVD`.

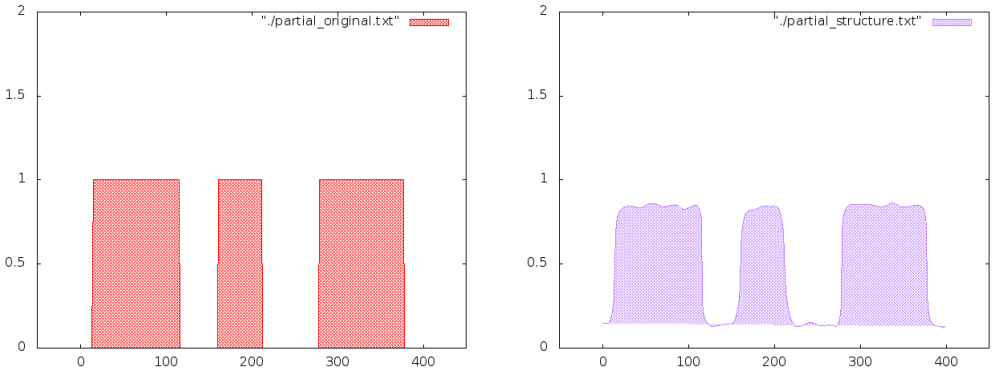
	Background	Quadrato	Triangolo	Cerchio
Originale	0	1	1	1
$TV - L^1$	0.12	0.88	0.85	0.87
$TV - L^2$	0.13	0.87	0.85	0.86
$TV - G$	0.13	0.83	0.81	0.83

Figura 3.17: Tabella delle medie dei valori dei pixel di ogni componente dell'immagine 3.9(c) prima e dopo la decomposizione con `ATVD`.

ni relative a 3.9(c) (ovvero la Fig. 3.18(a)) e prendiamo un piano parallelo al piano generato da un versore sull'asse z e un versore sull'asse y (le righe della matrice associata all'immagine). Facciamo in modo che tale piano passi per il diametro del cerchio rappresentato. Quello che vogliamo osservare è il grafico bidimensionale relativo alla sezione così ottenuta (vedi Fig. 3.18). In questi grafici appare molto chiaramente come tutti gli algoritmi tendano ad alzare i valori bassi e ad abbassare i valori alti e come i due algoritmi relativi alla norma G e alla norma L^1 tendano a smussare i bordi, anche se di poco. Notiamo che anche l'algoritmo $TV - L^1$ presenta delle irregolarità nelle superfici degli oggetti, così come l'algoritmo $TV - L^2$, anche se di natura più "dolce" rispetto alle variazioni di quest'ultimo.

3.3 Immagini astronomiche simulate

In questa sezione prenderemo in esame una simulazione molto accurata di un'immagine astronomica. Lo scopo è quello di effettuare una decomposizione tale che la parte struttura generata sia il più possibile uguale all'immagine priva di rumore 3.25(a). Anche in questa sezione sfrutteremo l'algoritmo di correlazione per selezionare automaticamente il parametro λ da utilizzare nella decomposizione. Vedremo anche come selezionare al meglio il parametro α (ovvero il parametro che regola la somiglianza richiesta tra la somma di struttura e texture e l'immagine con rumore, vedi 2.4) nel metodo $TV - G$ per ottenere risultati il più possibile accurati.



(a) Originale

(b) $TV - L^1$

(c) $TV - L^2$

(d) $TV - G$

Figura 3.18: Sezione delle immagini prodotte da `ATVD` e relative all'immagine 3.9(c).

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/geomnoise.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/geom.fits       Tempo -----------------> 14.853027
tau= 0.12                                       imin=  78
eps= 0.001                                      Lambda= 0.269722
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 69.736328
alpha= 1                                        Root-mean-square deviation
metodo= TVG                                     RMSE ------------------> 0.134987
statistica= SI                                  Peak-Signal to noise ratio
autolambda= SI                                  PSNR ------------------> 17.394170
Salva in= ./PROVE                               Coefficiente di variazione
Nome= CON_PARAMETRI                             CV ------------------> 0.307657
ni= 400                                         Structure Similarity Index
nj= 400                                         SSIM ------------------> 0.356429
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.890016
```

Figura 3.19: `Report` di 3.9(a) con l'algoritmo A^2BC $(TV - G)$

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/geomnoise.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/geom.fits       Tempo -----------------> 14.521729
tau= 0.12                                       imin=  78
eps= 0.001                                      Lambda= 0.269722
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 0.905273
metodo= TVL2                                    Root-mean-square deviation
statistica= SI                                  RMSE ------------------> 0.115520
autolambda= SI                                  Peak-Signal to noise ratio
Salva in= ./PROVE                               PSNR ------------------> 18.746855
Nome= CON_PARAMETRI                             Coefficiente di variazione
ni= 400                                         CV ------------------> 0.263289
nj= 400                                         Structure Similarity Index
                                                SSIM ------------------> 0.349290
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.910000
```

Figura 3.20: `Report` di 3.9(a) con l'algoritmo di Chambolle $(TV - L^2)$

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/geomnoise.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/geom.fits       Tempo -----------------> 32.170898
tau= 0.12                                       imin=  88
eps= 0.001                                      Lambda= 0.0940461
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 1.531738
alpha= 0.1                                      Root-mean-square deviation
metodo= TVL1                                    RMSE ------------------> 0.117618
statistica= SI                                  Peak-Signal to noise ratio
autolambda= SI                                  PSNR ------------------> 18.590548
Salva in= ./PROVE                               Coefficiente di variazione
Nome= CON_PARAMETRI                             CV ------------------> 0.268070
ni= 400                                         Structure Similarity Index
nj= 400                                         SSIM ------------------> 0.367406
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.910016
```

Figura 3.21: `Report` di 3.9(a) con l'algoritmo $Soft-Thresholding$ $(TV-L^1)$

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/secnoise.fits           Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/section.fits    Tempo -----------------> 14.693604
tau= 0.12                                       imin=  83
eps= 0.001                                      Lambda= 0.159268
eps2= 0.001                                     Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 78.791504
alpha= 1                                        Root-mean-square deviation
metodo= TVG                                     RMSE ------------------> 0.147588
statistica= SI                                  Peak-Signal to noise ratio
autolambda= SI                                  PSNR ------------------> 16.619003
Salva in= ./PROVE                               Coefficiente di variazione
Nome= CON_PARAMETRI                             CV ------------------> 0.388624
ni= 400                                         Structure Similarity Index
nj= 400                                         SSIM ------------------> 0.162007
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.870016
```

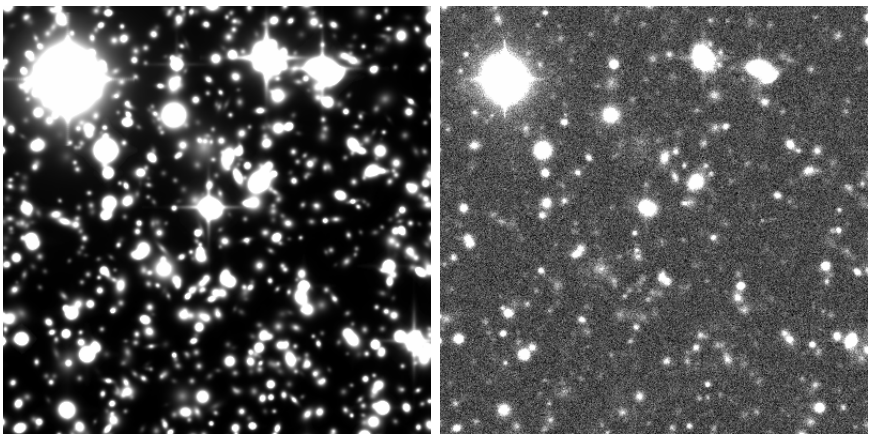
Figura 3.22: `Report` di 3.9(c) con l'algoritmo A^2BC $(TV-G)$

```
Astro Total Variation Denoiser

Image (Noised):  ./FITS/secnoise.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/section.fits   Tempo -----------------> 13.479004
tau= 0.12                                      imin=  83
eps= 0.001                                     Lambda= 0.159268
eps2= 0.001                                    Tempo impiegato per la soluzione
numero di threads= 4                           Tempo -----------------> 1.080322
metodo= TVL2                                   Root-mean-square deviation
statistica= SI                                 RMSE ------------------> 0.132193
autolambda= SI                                 Peak-Signal to noise ratio
Salva in= ./PROVE                              PSNR ------------------> 17.575824
Nome= CON_PARAMETRI                            Coefficiente di variazione
ni= 400                                        CV ------------------> 0.348088
nj= 400                                        Structure Similarity Index
                                               SSIM ------------------> 0.168742
                                               Tempo impiegato per la scrittura
                                               Tempo -----------------> 1.130000
```

Figura 3.23: `Report` di 3.9(c) con l'algoritmo di Chambolle ($TV - L^2$)

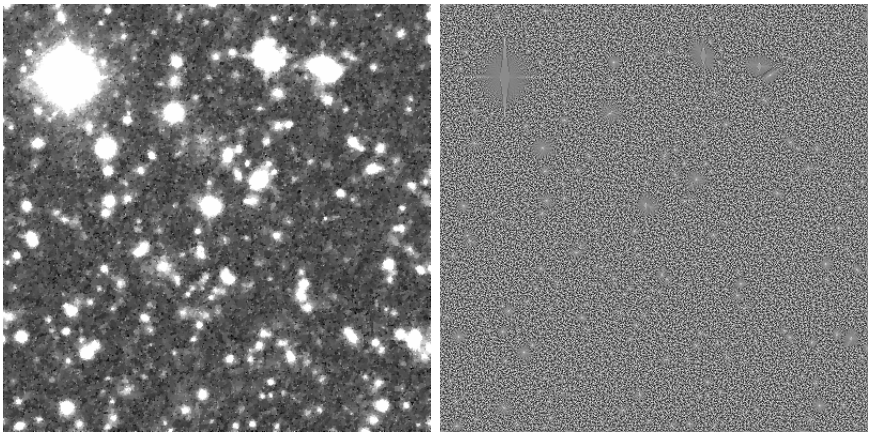
```
                      Astro Total Variation Denoiser

Image (Noised):  ./FITS/secnoise.fits          Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/section.fits   Tempo -----------------> 31.853271
tau= 0.12                                      imin=  83
eps= 0.001                                     Lambda= 0.159268
eps2= 0.001                                    Tempo impiegato per la soluzione
numero di threads= 4                           Tempo -----------------> 0.380615
alpha= 0.01                                    Root-mean-square deviation
metodo= TVL1                                   RMSE ------------------> 0.148152
statistica= SI                                 Peak-Signal to noise ratio
autolambda= SI                                 PSNR ------------------> 16.585841
Salva in= ./PROVE                              Coefficiente di variazione
Nome= CON_PARAMETRI                            CV ------------------> 0.390111
ni= 400                                        Structure Similarity Index
nj= 400                                        SSIM ------------------> 0.155927
                                               Tempo impiegato per la scrittura
                                               Tempo -----------------> 0.880000
```

Figura 3.24: `Report` di 3.9(c) con l'algoritmo $Soft-Thresholding$ ($TV-L^1$)



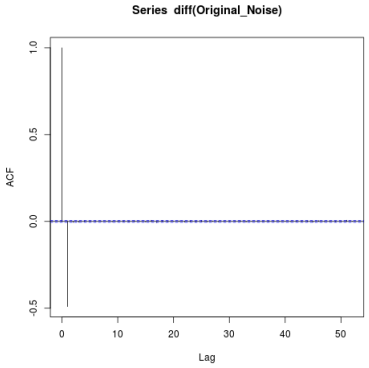
(a) `Original.fits` (b) `Noise.fits`

Figura 3.25: Simulazione di un'immagine astronomica.

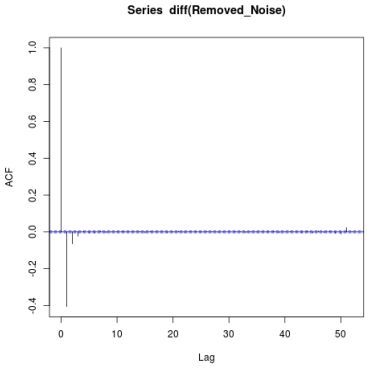


(a) `Structural.fits` (b) `Texture.fits`

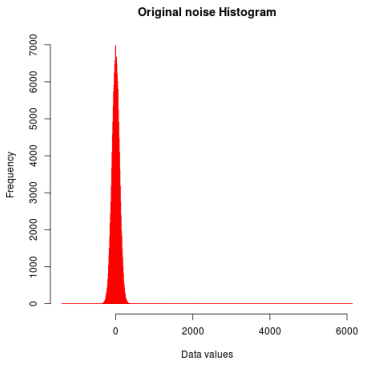
Figura 3.26: Risultati della decomposizione effettuata con il metodo $TV-L^2$.



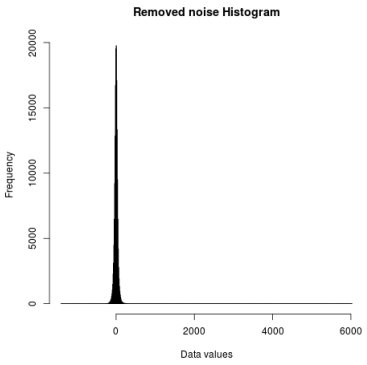
(a) ACF differenze rumore originale



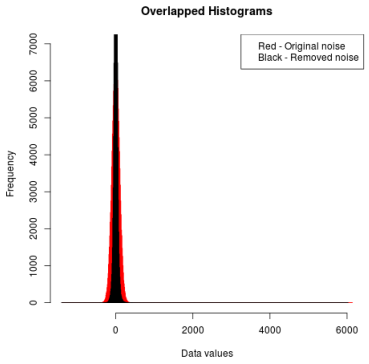
(b) ACF differenze rumore rimosso



(c) Istogramma rumore originale



(d) Istogramma rumore rimosso



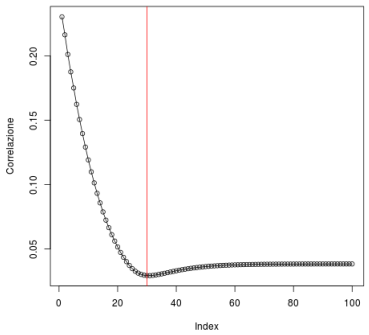
(e) Istogrammi sovrapposti

Figura 3.27: Analisi grafica (effettuata con il software `R`) del rumore rimosso dal programma con il metodo $TV - L^2$ e sovrapposizione con istogramma relativo al rumore originale.

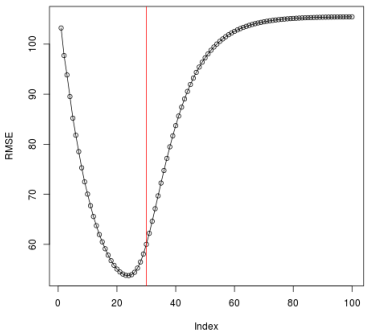
3.3.1 Risoluzione con metodo $TV - L^2$

Se eseguiamo `ATVD` su questa simulazione astronomica utilizzando il metodo $TV - L^2$ otterremo i risultati proposti nella Fig. 3.28(a). Ad una prima analisi il rumore presente sembra essere decisamente diminuito: osservando i grafici relativi all'analisi del rumore rimosso (Fig. 3.27) e confrontandolo con il rumore originale possiamo sicuramente affermare che non solo il rumore rimosso è stato generato da una sorgente gaussiana (con molta probabilità) e presenta anche un buon grado di scorrelazione, ma anche che tale rumore si sovrappone bene al rumore originale.

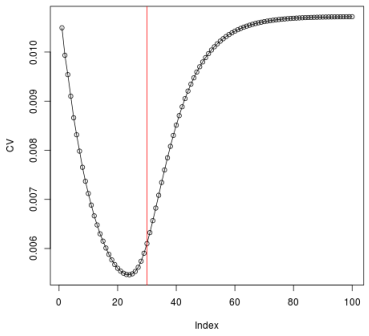
Sfortunatamente però l'analisi dei quattro test di somiglianza non va bene come negli altri casi (Fig. 3.28): sembrerebbe quasi che il principio di minima correlazione non sia valido nel caso di immagini astronomiche con un elevato numero di oggetti. Abbiamo verificato che tale comportamento si ripete per tutte le simulazioni di immagini astronomiche con molti oggetti prese in considerazione nel corso del nostro lavoro. L'algoritmo di correlazione fornisce comunque un risultato utile per avvicinarci alla miglior decomposizione. La causa di questa differenza rispetto alle simulazioni non astronomiche discusse in precedenza è probabilmente da attribuirsi alla presenza di sorgenti deboli completamente nascoste dal livello di rumore presente e che non vengono individuate come parte struttura. Stiamo attualmente indagando in modo più approfondito questa problematica nella decomposizione di immagini astronomiche. In particolare per capire se è possibile superare questo limite dell'algoritmo di correlazione adattandolo meglio alle caratteristiche delle immagini astronomiche, o se migliori risultati possono essere ottenuti con altre tecniche di selezione del parametro λ (e.g. [4, 5, 7, 8, 11, 15]).



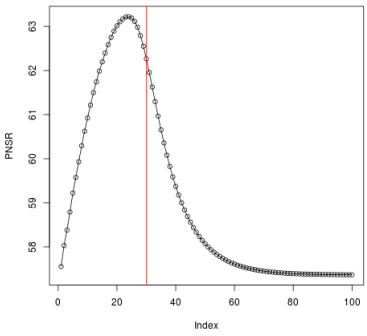
(a) `Correlazione.png`



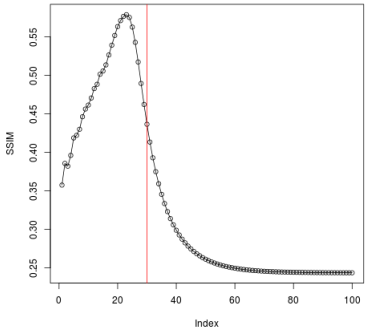
(b) `RMSE.png`



(c) `CV.png`



(d) `PNSR.png`



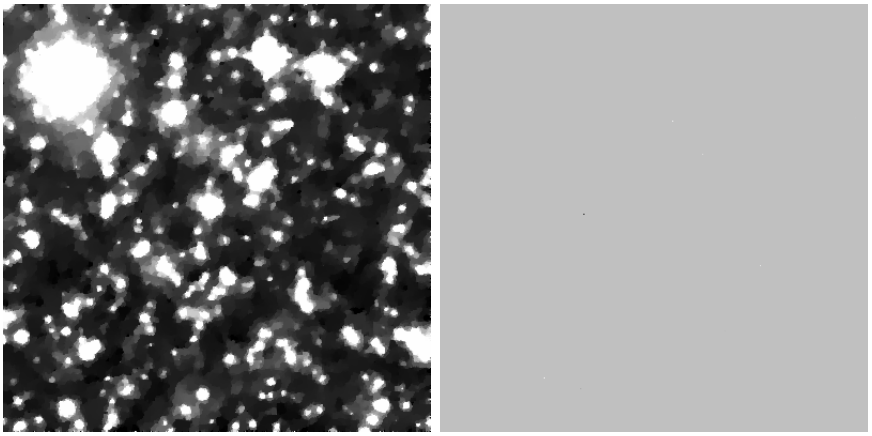
(e) `SSIM.png`

Figura 3.28: Grafici dei risultati dei test di somiglianza al variare del parametro λ.

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/400.fits              Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/400c.fits     Tempo -----------------> 22.735352
tau= 0.12                                     imin=  30
eps= 0.001                                    Lambda= 42.3912
eps2= 1e-05                                   Tempo impiegato per la soluzione
numero di threads= 4                          Tempo -----------------> 26.615234
metodo= TVL2                                  Root-mean-square deviation
statistica= SI                                RMSE ------------------> 62.206176
autolambda= SI                                Peak-Signal to noise ratio
Salva in= ./PROVE                             PSNR ------------------> 61.954079
Nome= CON_PARAMETRI                           Coefficiente di variazione
ni= 400                                       CV ------------------> 0.006324
nj= 400                                       Structure Similarity Index
                                              SSIM ------------------> 0.413407
                                              Tempo impiegato per la scrittura
                                              Tempo -----------------> 0.220000
```

Figura 3.29: `Report` di 3.25(a) con l'algoritmo di Chambolle ($TV - L^2$)

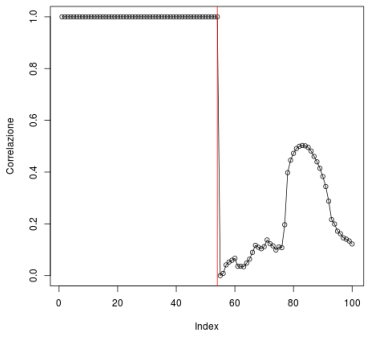
3.3.2 Risoluzione con metodo $TV - L^1$



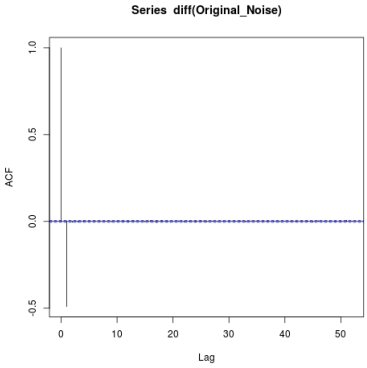
(a) `Structural.fits` (b) `Texture.fits`

Figura 3.30: Risultati della decomposizione effettuata con il metodo $TV - L^1$.

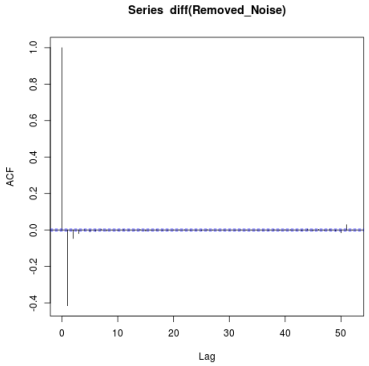
Il metodo $TV - L^1$ restituisce i risultati presentati in Fig. 3.33. Anche in questo caso possiamo notare come molto rumore sia stato rimosso dal programma, producendo un output "migliore" del precedente metodo (com'è anche riportato dai test di somiglianza, vedi report in Fig. 3.29 e 3.32). L'analisi del rumore riportata in Fig. 3.31 produce risultati molto simili a quelli mostrati nella precedente sottosezione.



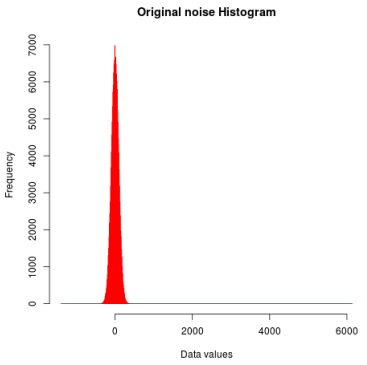
(a) `Correlazione.png`



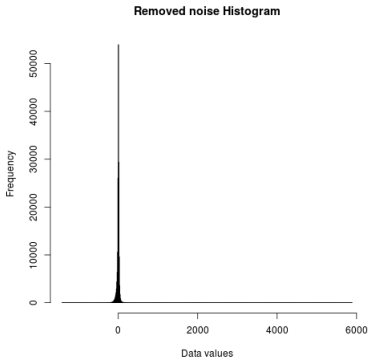
(b) ACF differenze rumore originale



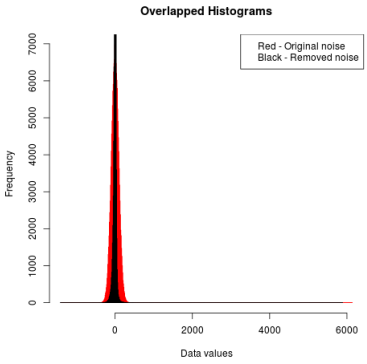
(c) ACF differenze rumore rimosso



(d) Istogramma rumore originale



(e) Istogramma rumore rimosso



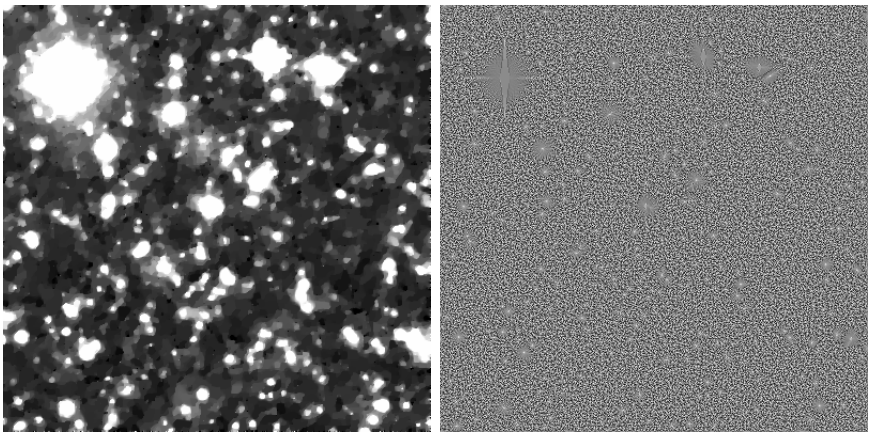
(f) Istogrammi sovrapposti

Figura 3.31: Grafico di correlazione, analisi grafica (effettuata con il software `R`) del rumore rimosso dal programma con il metodo $TV - L^1$ e sovrapposizione con istogramma relativo al rumore originale.

```
                        Astro Total Variation Denoiser

Image (Noised):  ./FITS/400.fits                Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/400c.fits       Tempo -----------------> 1395.351562
tau= 0.12                                       imin=  54
eps= 0.1                                        Lambda= 3.38139
eps2= 0.1                                       Tempo impiegato per la soluzione
numero di threads= 4                            Tempo -----------------> 4.796875
alpha= 100                                      Root-mean-square deviation
metodo= TVL1                                    RMSE ------------------> 53.863151
statistica= SI                                  Peak-Signal to noise ratio
autolambda= SI                                  PSNR ------------------> 63.204914
Salva in= ./PROVE                               Coefficiente di variazione
Nome= CON_PARAMETRI                             CV ------------------> 0.005475
ni= 400                                         Structure Similarity Index
nj= 400                                         SSIM ------------------> 0.580035
                                                Tempo impiegato per la scrittura
                                                Tempo -----------------> 0.200704
```

Figura 3.32: `Report` di 3.25(a) con l'algoritmo $Soft - Thresholding$ ($TV - L^1$)

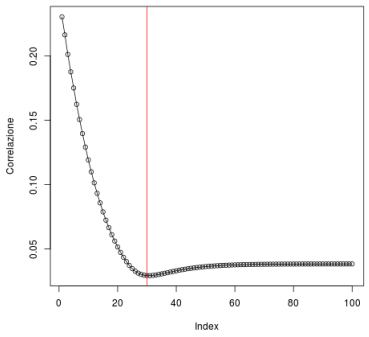
3.3.3 Risoluzione con metodo $TV - G$



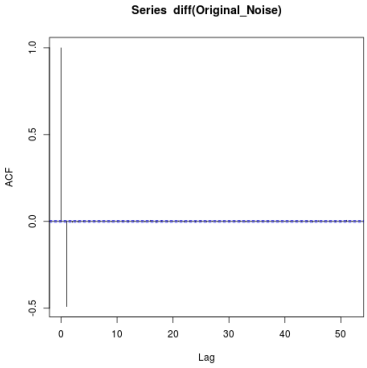
(a) `Structural.fits` (b) `Texture.fits`

Figura 3.33: Risultati della decomposizione effettuata con il metodo $TV - G$.

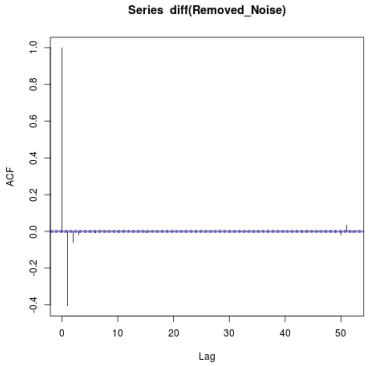
In questo caso, il metodo $TV - G$ produce i risultati migliori. Anche qui valgono le stesse considerazioni effettuate nella sottosezione 3.3.1.



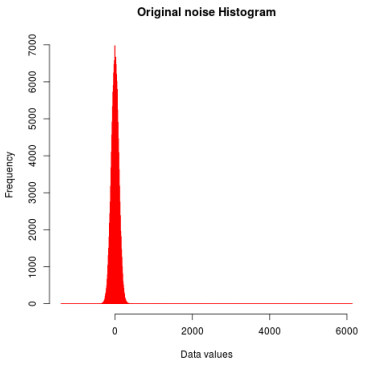
(a) `Correlazione.png`



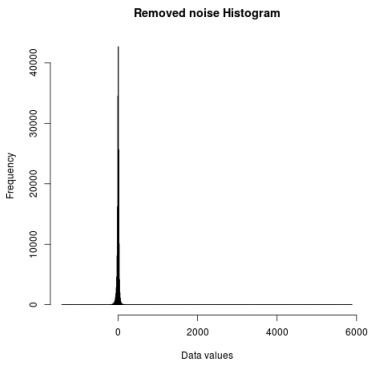
(b) ACF differenze rumore originale



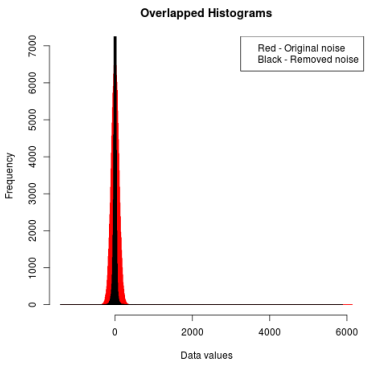
(c) ACF differenze rumore rimosso



(d) Istogramma rumore originale



(e) Istogramma rumore rimosso



(f) Istogrammi sovrapposti

Figura 3.34: Grafico di correlazione, analisi grafica (effettuata con il software `R`) del rumore rimosso dal programma con il metodo $TV-G$ e sovrapposizione con istogramma relativo al rumore originale.

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/400.fits              Tempo impiegato per correlazione
Original Image (Clean):  ./FITS/400c.fits     Tempo -----------------> 18.187500
tau= 0.12                                     imin=  30
eps= 0.001                                    Lambda= 42.3912
eps2= 1e-05                                   Tempo impiegato per la soluzione
numero di threads= 4                          Tempo -----------------> 658.851562
alpha= 42.3912                                Root-mean-square deviation
metodo= TVG                                   RMSE ------------------> 52.336755
statistica= SI                                Peak-Signal to noise ratio
autolambda= SI                                PSNR ------------------> 63.454613
Salva in= ./PROVE                             Coefficiente di variazione
Nome= CON_PARAMETRI                           CV ------------------> 0.005320
ni= 400                                       Structure Similarity Index
nj= 400                                       SSIM ------------------> 0.584364
                                              Tempo impiegato per la scrittura
                                              Tempo -----------------> 0.199936
```

Figura 3.35: `Report` di 3.25(a) con l'algoritmo A^2BC $(TV - G)$

3.3.4 Un confronto tra i risultati

Come possiamo notare esaminando i grafici proposti nella Fig. 3.28, in questo caso l'algoritmo di correlazione non produce i migliori risultati possibili, come era accaduto in precedenza. Essendo questa una simulazione però, possiamo comunque estrapolare il miglior parametro λ dai grafici della Fig. 3.28. In questa sottosezione mostreremo un confronto tra l'immagine affetta da rumore, l'immagine originale completamente priva di rumore e i risultati dei nostri test, nei quali inseriremo anche la decomposizione effettuata dal programma sfruttando il miglior parametro λ^* (ovvero il parametro che massimizza l'indice `SSIM`) secondo il metodo $TV - G$ (figure 3.36, 3.37 e 3.38). Notiamo come molti oggetti nascosti vengono comunque portati in evidenza da ogni metodo. In ultimo proporremo il report della prova effettuata con il parametro λ^*.

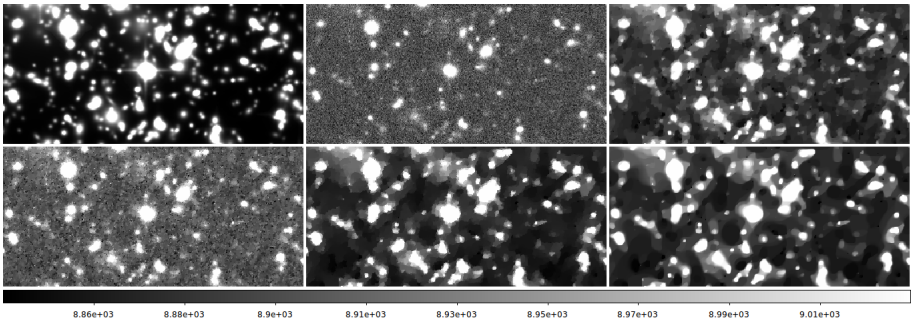


Figura 3.36: Confronto tra le immagini "ripulite" prodotte da `ATVD`. In alto a sinistra troviamo l'immagine originale priva di rumore, a seguire da sinistra a destra possiamo osservare l'immagine affetta da rumore e le decomposizioni effettuate con i metodi $TV - L^1$, $TV - L^2$ e $TV - G$. In ultimo abbiamo l'immagine decomposta con il metodo $TV - G$ utilizzando il parametro λ^*.

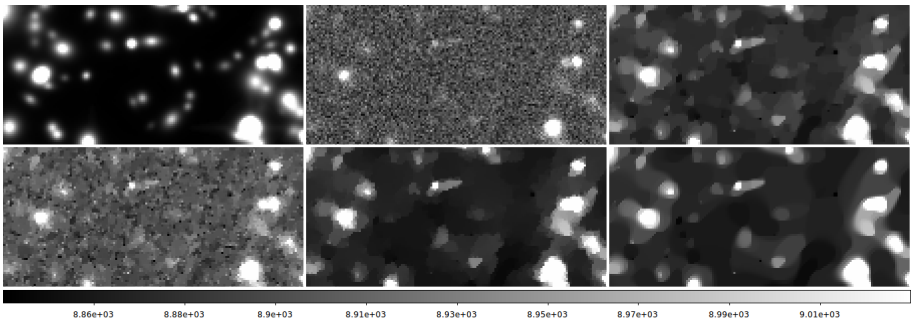


Figura 3.37: Confronto tra le immagini "ripulite" prodotte da `ATVD` (particolare).

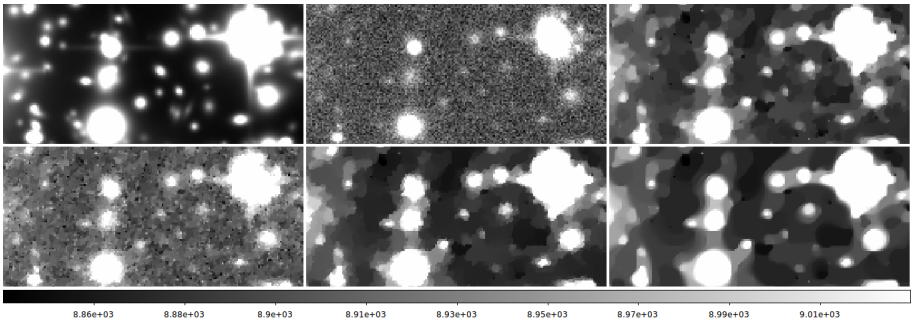


Figura 3.38: Confronto tra le immagini "ripulite" prodotte da `ATVD` (particolare).

```
                    Astro Total Variation Denoiser

Image (Noised):  ./FITS/400.fits              Tempo impiegato per la soluzione
Original Image (Clean):  ./FITS/400c.fits     Tempo -----------------> 5958.257812
tau= 0.12                                     Root-mean-square deviation
eps= 0.001                                    RMSE ------------------> 52.757042
eps2= 1e-05                                   Peak-Signal to noise ratio
numero di threads= 4                          PSNR ------------------> 63.385140
alpha= 1                                      Coefficiente di variazione
metodo= TVG                                   CV ------------------> 0.005363
statistica= SI                                Structure Similarity Index
autolambda= NO                                SSIM ------------------> 0.593908
Lambda= 88.6294                               Tempo impiegato per la scrittura
Salva in= ./PROVE                             Tempo -----------------> 0.196608
Nome= CON_PARAMETRI
ni= 400
nj= 400
Lambda= 88.6294
```

Figura 3.39: `Report` di 3.9(c) con l'algoritmo A^2BC $(TV - G)$ utilizzando λ^*

3.3.5 Selezione di sorgenti sull'immagine struttura

Dagli esempi in Fig. 3.36, 3.37 e 3.38, risulta che nell'immagine struttura è visibile un maggior numero di sorgenti rispetto all'immagine di input, mentre le sorgenti presenti in entrambe appaiono molto più nitide. Abbiamo verificato quantitativamente tale risultato eseguendo la selezione delle sorgenti tramite l'utilizzo del codice `SExtractor`([23]) comunemente usato per la "detection" di oggetti in immagini astronomiche. In Fig.3.40 è mostrata la distribuzione del numero di oggetti presenti nelle due immagini (l'originale in input e quella "denoised", cioé l'immagine Struttura) in funzione della "magnitudine" (Mag $\propto -2.5 \log$(flusso), dunque a magnitudini maggiori corrispondono le sorgenti con flusso minore). La rimozione del rumore effettuata da ATVD permette di individuare sull'immagine struttura un maggior numero di sorgenti deboli di quante possono essere selezionate sull'immagine di input: *questo è un risultato estremamente promettente per l'utilizzo delle tecniche di decomposizione nell'ambito dell'analisi scientifica di sorgenti astrofisiche.* In Fig.3.41 è riportato il rapporto S/N in funzione della magnitudine per le sorgenti selezionate nelle due immagini, a conferma del fatto che la rimozione del rumore è stata efficace rendendo più nitide anche le sorgenti brillanti già visibili nell'immagine di input.

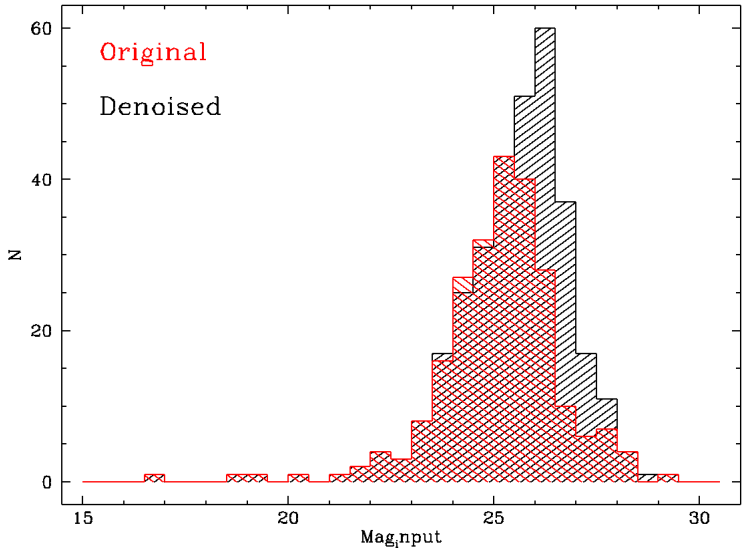


Figura 3.40: Distribuzione del numero di oggetti presenti nell' immagine originale in input e in quella stuttura in output di ATVD ("denoised"), in funzione della "magnitudine" (Mag $\propto -2.5 \log(\text{flusso})$).

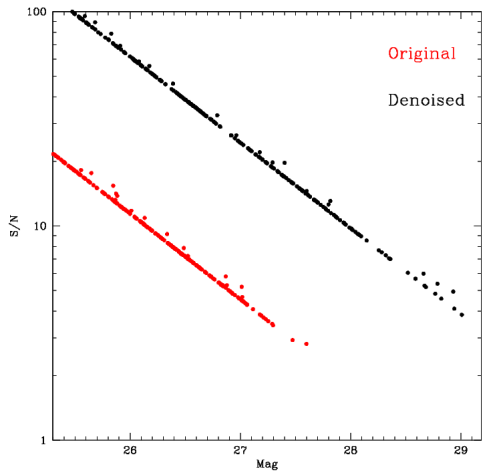


Figura 3.41: Rapporto S/N in funzione della magnitudine nelle immagini di input e output.

3.4 Decomposizione di immagini astronomiche reali

Concludiamo questo lavoro con l'applicazione di `ATVD` a immagini astronomiche reali, delle quali ovviamente non si possiede alcuna "versione originale" totalmente priva di rumore per eseguire test di somiglianza come in modalità simulazione di ATVD.

In ogni caso potremmo continuare ad usare l'algoritmo di correlazione per la selezione automatica del parametro λ. Uno dei punti di forza di questo algoritmo è infatti la non dipendenza dalla disponibilità di altre immagini oltre che da quelle prodotte dal programma stesso. In questa sezione prenderemo in esame una reale osservazione di una porzione di cielo ottenuta mediante il telescopio `HUBBLE` (Fig. 3.42). Nello specifico, si tratta di una osservazione infrarossa (lunghezza d'onda=1.6 μm) del campo "Hubble Ultra-Deep Field" eseguita nel corso del programma osservativo CANDELS([24]). Di tale campo abbiamo a disposizione due immagini, denominate "GS-DEEP" e "GS-HUDF" ottenute con diverso tempo di esposizione. In particolare, GS-HUDF è stata ottenuta con un tempo di esposizione molto più lungo rispetto a GS-DEEP e in essa dunque risultano individuabili sorgenti molto più deboli.

Come primo passo, per poter valutare l'efficacia del nostro algoritmo di decomposizione su immagini astronomiche reali utilizzeremo ATVD su GS-DEEP e verifichiamo se le sorgenti individuate sull'immagine struttura siano presenti o meno nell'immagine GS-HUDF. In Fig. 3.43 possiamo osservare il grafico prodotto dall'algoritmo di correlazione. La Fig. 3.45 mostra invece l'andamento dei test statistici effettuati sull'immagine decomposta secondo il parametro λ suggerito dall'algoritmo stesso. In Fig. 3.44 possiamo invece osservare le decomposizioni proposte dal programma secondo il metodo $TV - L^2$ e secondo il metodo $TV - G$. Nella Fig. 3.46 presentiamo le tre immagini GS-DEEP, GS-HUDF e la parte struttura di GS-DEEP. La Fig. 3.47 rappresenta un particolare della Fig. 3.46; le altre due figure invece (3.48 e 3.49) testimoniano l'efficacia dell'algoritmo: sorgenti dapprima completamente coperte dal rumore diventano visibili nell'immagine decomposta. Purtroppo però non tutte le sorgenti visibili sulla parte struttura di

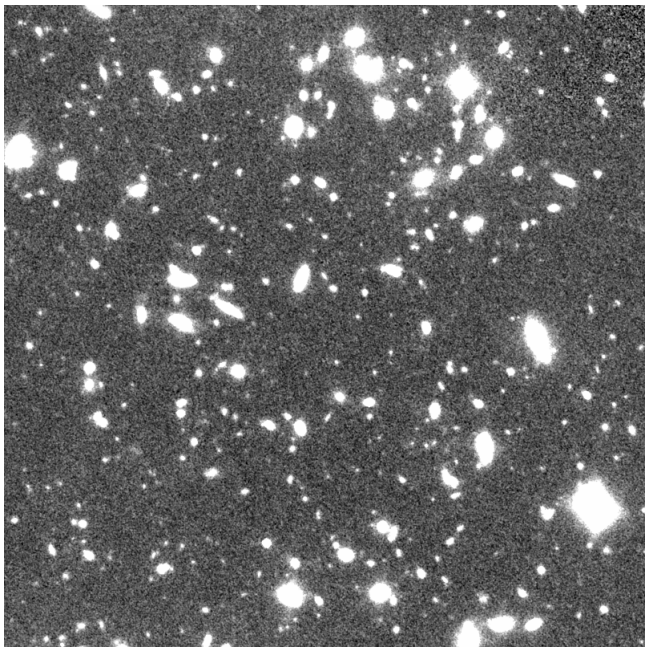


Figura 3.42: Immagine astronomica reale presa dal telescopio HUBBLE.

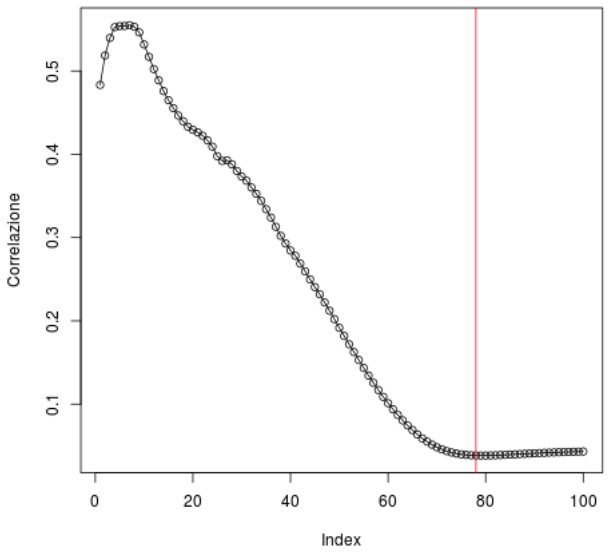
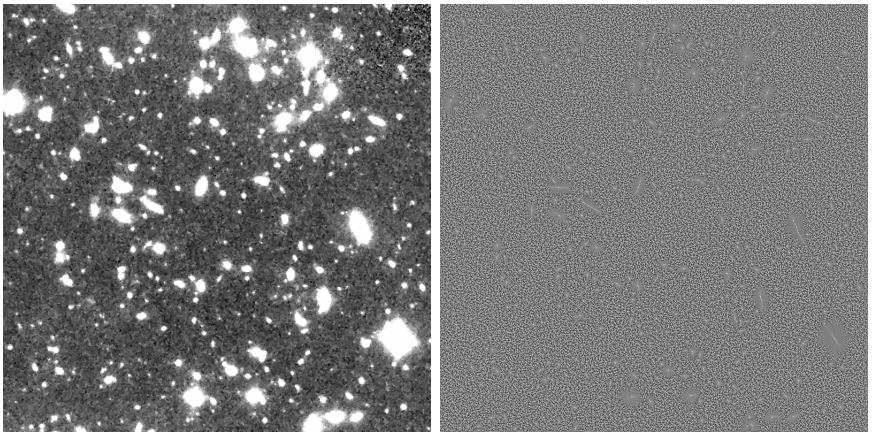


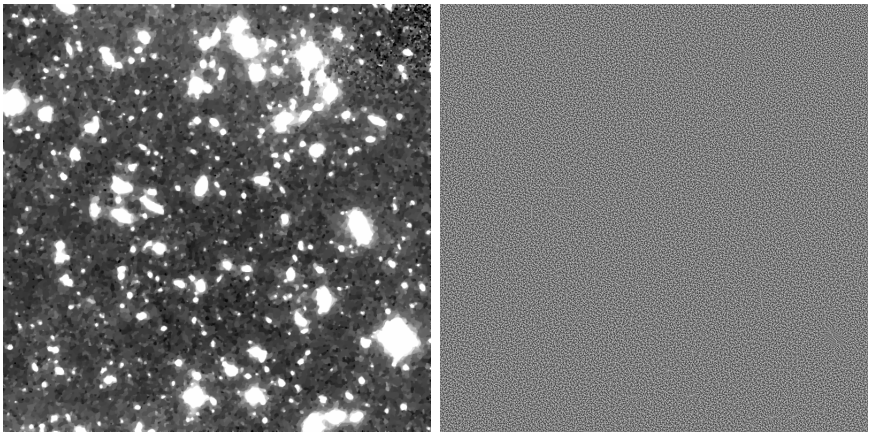
Figura 3.43: Grafico di correlazione relativo all'immagine 3.42.

GS-DEEP risultano confermate in GS-HUDF. Zone dell'immagine di input in cui sono presenti dei difetti o la presenza di rumore correlato (questa può essere dovuta a difetti strumentali o imprecisioni nell'acquisizione delle immagini) generano delle sorgenti spurie nell'immagine stuttura. Un esempio è mostrato in Fig. 3.50. La soluzione di questo problema sarà oggetto di futuro lavoro.

Analogamente a quanto fatto per le simulazioni discusse nel Par. 3.3.5 valutiamo quantitativamente il numero di sorgenti in funzione della magnitudine, e il rapporto S/N degli oggetti individuati dal codice `SExtractor` sulla parte struttura di GS-DEEP. Un paragone con l'analisi tramite `SExtractor` dell'immagine GS-DEEP di input e dell'immagine GS-HUDF ci permette di misurare il numero di sorgenti la cui individuazione è resa possibile dalla decomposizione dell'immagine (cioè sorgenti presenti anche in GS-HUDF), e il numero di sorgenti spurie (individuate sulla parte struttura di GS-DEEP ma non presenti in GS-HUDF). Il risultato è mostrato in Fig. 3.53: come si può vedere, un gran numero di sorgenti invisibili nell'immagine di input possono essere individuate grazie all'immagine struttura generata da ATVD (istogramma verde). Come discusso in precedenza è presente anche un piccolo numero di sorgenti spurie (in magenta), mentre un paio di sorgenti reali non sono state individuate. Analogamente a quanto mostrato nelle simulazioni, il rapporto S/N degli oggetti individuati risulta molto maggiore (Fig.3.54).

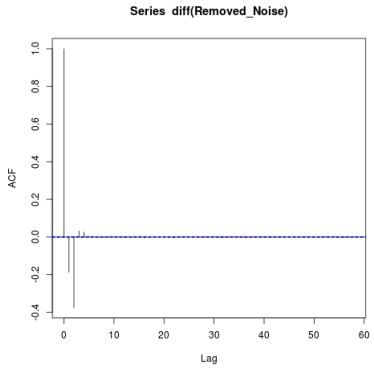


(a) $TV - L^2$ (b)

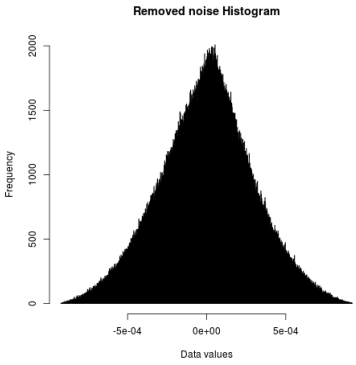


(c) $TV - G$ (d)

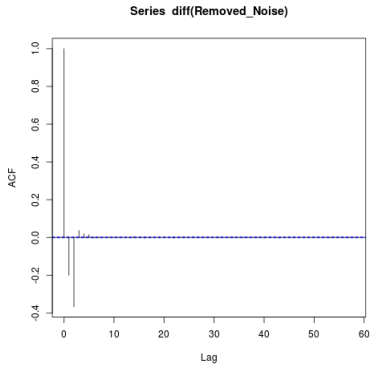
Figura 3.44: Decomposizioni di **??** effettuate da `ATVD`



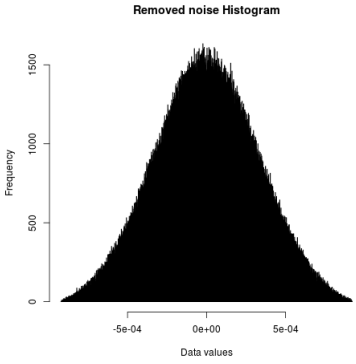
(a) ACF differenze rumore rimosso ($TV - L^2$)



(b) Istogramma rumore rimosso ($TV - L^2$)



(c) ACF differenze rumore rimosso ($TV - G$)



(d) Istogramma rumore rimosso ($TV - G$)

Figura 3.45: Analisi grafica (effettuata con il software `R`) del rumore rimosso dal programma

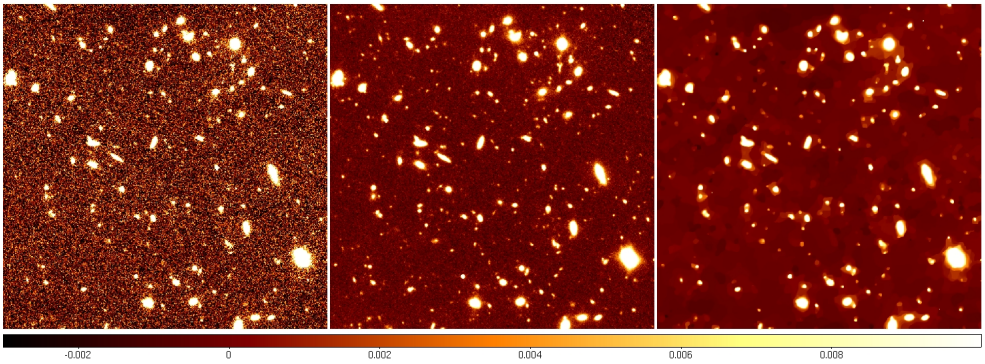


Figura 3.46: Confronto tra l'immagine GS-DEEP (quella su cui abbiamo effettuato la decomposizione), GS-HUDF (la stessa zona di cielo osservata con tempo di esposizione maggiore) e la parte struttura dell'immagine decomposta GS-DEEP con il metodo $TV - G$.

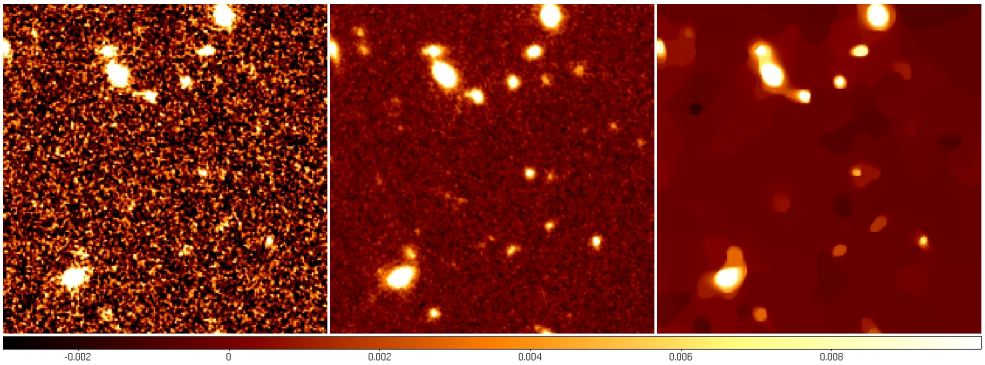


Figura 3.47: Particolare di 3.46.

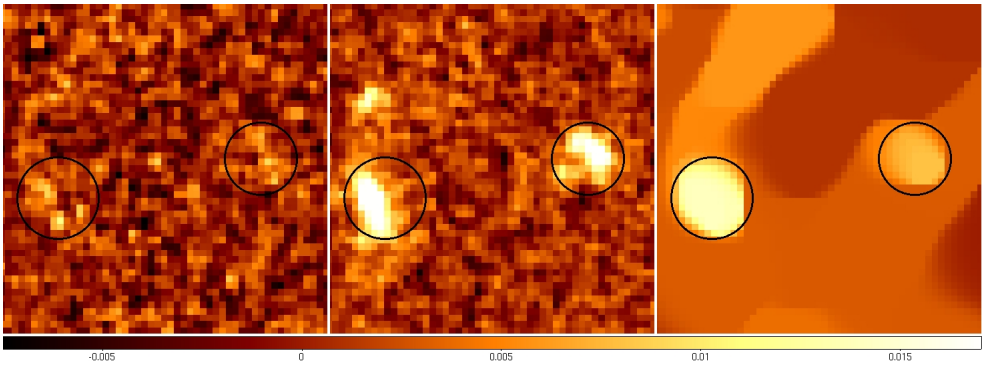


Figura 3.48: Particolare di 3.46. Sono evidenziate le sorgenti che inizialmente erano completamente ricoperti dal rumore.

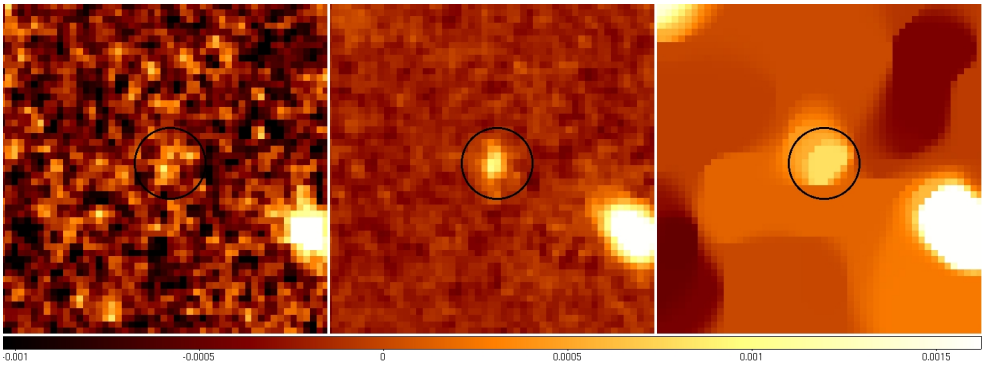


Figura 3.49: Come 3.48.

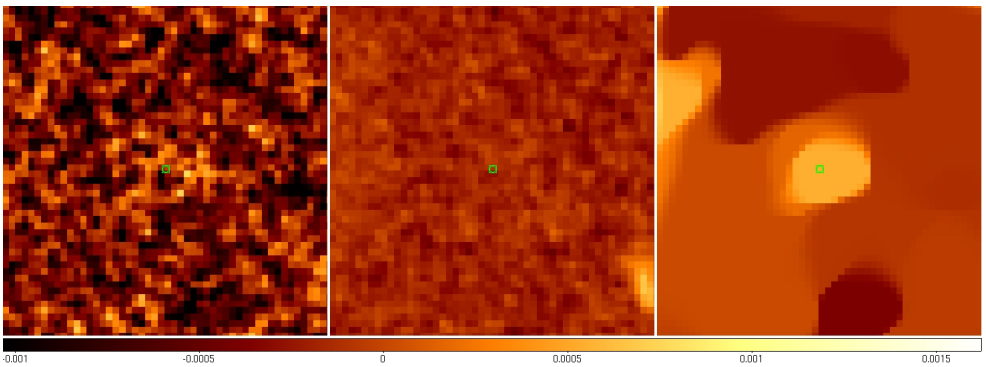


Figura 3.50: Particolare di 3.46. La piccola circonferenza verde indica un elemento riconosciuto come significativo dalla decomposizione ma che non risulta essere un vero corpo celeste.

```
Astro Total Variation Denoiser

Image (Noised):  ./FITS/SALL.fits
tau= 0.12
eps= 0.001
eps2= 1e-05
numero di threads= 4
alpha= 0.000269722
metodo= TVG
statistica= SI
autolambda= SI
Salva in= ./PROVE
Nome= CON_PARAMETRI
ni= 800
nj= 800
Tempo impiegato per correlazione
Tempo -----------------> 142.468750
imin=  78
Lambda= 0.000269722
Tempo impiegato per la soluzione
Tempo -----------------> 784.734375
Tempo impiegato per la scrittura
Tempo -----------------> 0.310016
```

Figura 3.51: `Report` di 3.9(c) con l'algoritmo A^2BC $(TV - G)$

```
Astro Total Variation Denoiser

Image (Noised):  ./FITS/SALL.fits
tau= 0.12
eps= 0.001
eps2= 1e-05
numero di threads= 4
metodo= TVL2
statistica= SI
autolambda= SI
Salva in= ./PROVE
Nome= CON_PARAMETRI
ni= 800
nj= 800
Tempo impiegato per correlazione
Tempo -----------------> 152.070312
imin=  78
Lambda= 0.000269722
Tempo impiegato per la soluzione
Tempo -----------------> 184.726562
Tempo impiegato per la scrittura
Tempo -----------------> 0.310016
```

Figura 3.52: `Report` di 3.9(c) con l'algoritmo di Chambolle $(TV - L^2)$

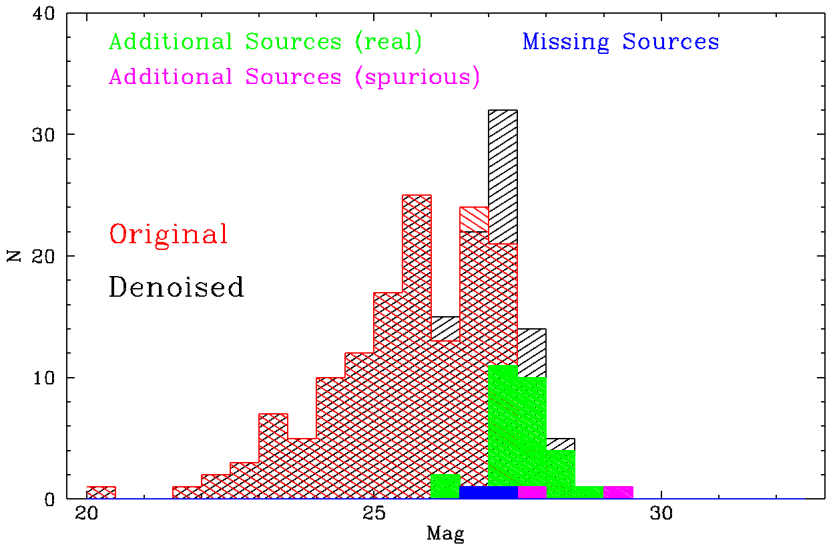


Figura 3.53: Numero di sorgenti in funzione della magnitudine (Mag $\propto -2.5\log(\text{flusso})$) individuate sull'immagine di input (istogramma rosso) e sulla parte struttura di GS-DEEP (nero). Gran parte delle sorgenti aggiuntive sono confermate come reali da un'analisi dell'immagine GS-HUDF ottenuta con tempo di esposizione maggiore rispetto a GS-DEEP.

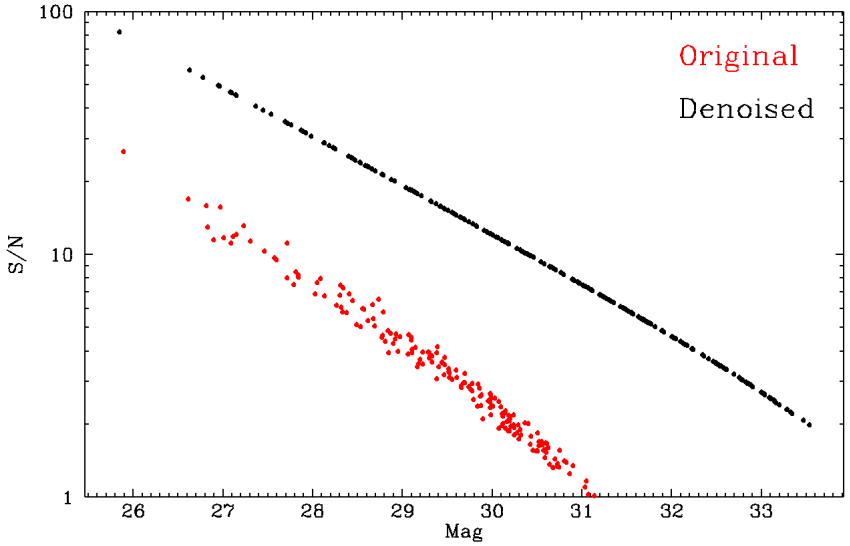


Figura 3.54: Rapporto S/N delle sorgenti nell'immagine di input e nell'immagine struttura in output di ATVD.

3.4.1 Individuazione di nuove sorgenti nell'immagine ultra-fonda GS-HUDF

Confortati dai risultati descritti in precedenza abbiamo decomposto tramite l'algoritmo $TV - G$ l'immagine GS-HUDF e cercato sorgenti aggiuntive che non fossero visibili in precedenza a causa del rumore. Analogamente a quanto mostrato nei precedenti paragrafi, riportiamo in Fig.3.55 i conteggi delle sorgenti trovate sull'immagine struttura in funzione della magnitudine paragonate alla distribuzione delle sorgenti nell'immagine di input. In verde il sottoinsieme delle sorgenti aggiuntive. *La decomposizione permette di estendere a magnitudini maggiori (dunque flussi più bassi) la selezione degli oggetti celesti.*

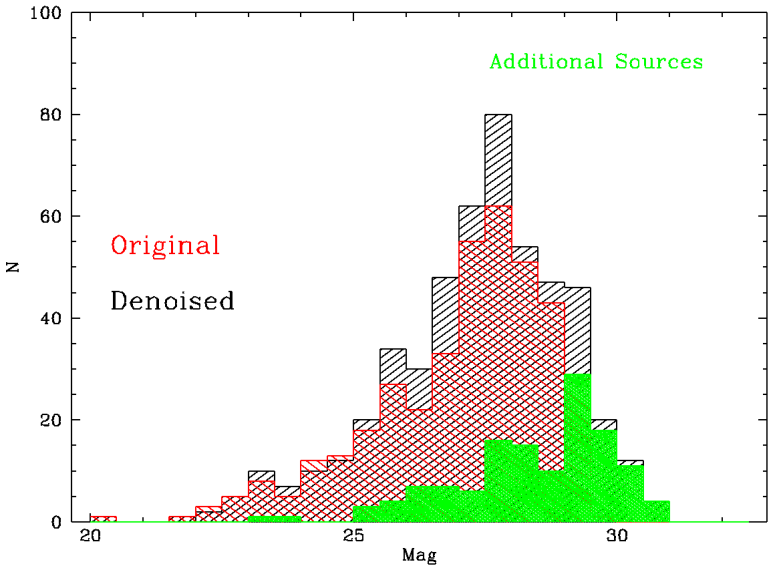


Figura 3.55: Numero di sorgenti in funzione della magnitudine (Mag $\propto -2.5\log$(flusso)) individuate sull'immagine di input (istogramma rosso) e sulla parte struttura di GS-HUDF (nero). Il numero di sorgenti aggiuntive trovate su quest'ultima è indicato dall'istogramma verde.

Abbiamo ispezionato visualmente le sorgenti aggiuntive trovate e cercato se fossero presenti anche in immagini delle stessa zona di cielo ottenute a diverse lunghezze d'onda (λ= 0.4, 0.9 e 1.2 μm). Esempi sono riportati in Fig. 3.57 e 3.58. È interessante notare che parte delle sorgenti trovate sono

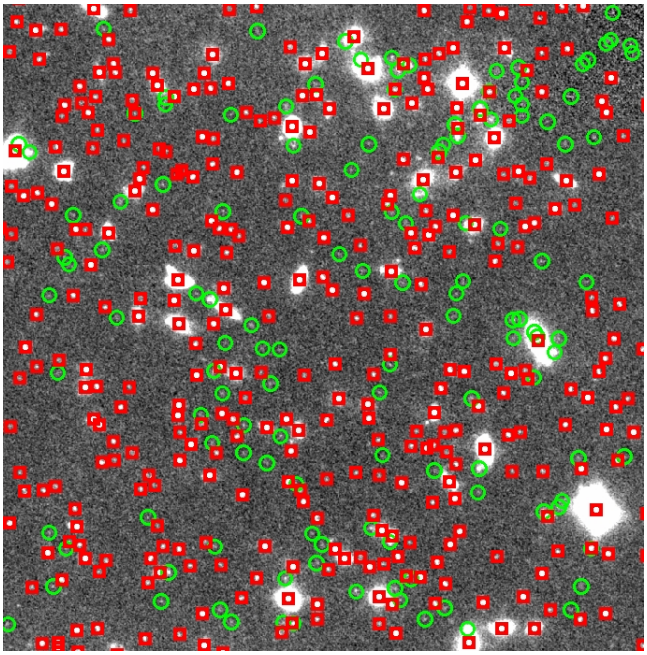


Figura 3.56: Sorgenti trovate sulla parte struttura di GS-HUDF: sorgenti note (rosso) e aggiuntive (verde).

visibili, anche se deboli, nelle immagini ad altre lunghezze d'onda, mentre risultano pressochè invisibili nell'immagine di input. **Questo dimostra che l'utilizzo delle tecniche di decomposizione implementate in ATVD permette di migliorare la nostra capacità di selezione di oggetti celesti estremamente deboli**.

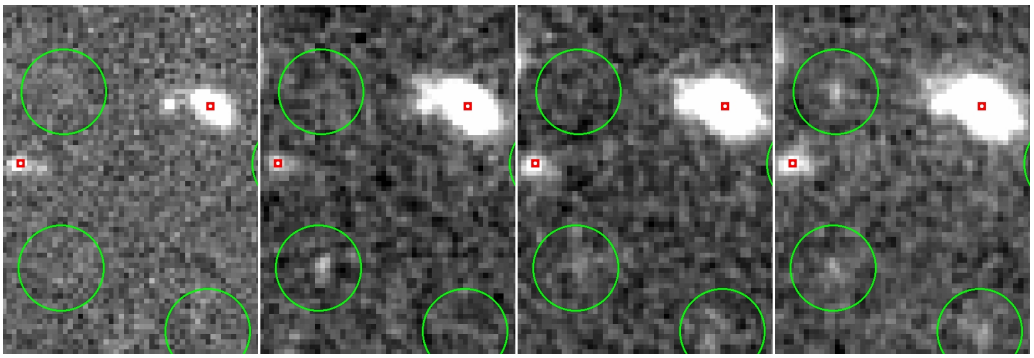


Figura 3.57: Dettaglio di Fig. 3.56. Da sinistra a destra: immagini del campo HUDF ottenute a lunghezza d'onda λ= 0.4, 0.9, 1.2 e 1.6 μm. Evidenziate in verde: sorgenti aggiuntive trovate sulla parte struttura dell'immagine GS-HUDF a 1.6 μm (decomposizione $TV - G$). In rosso: sorgenti già note in precedenza.

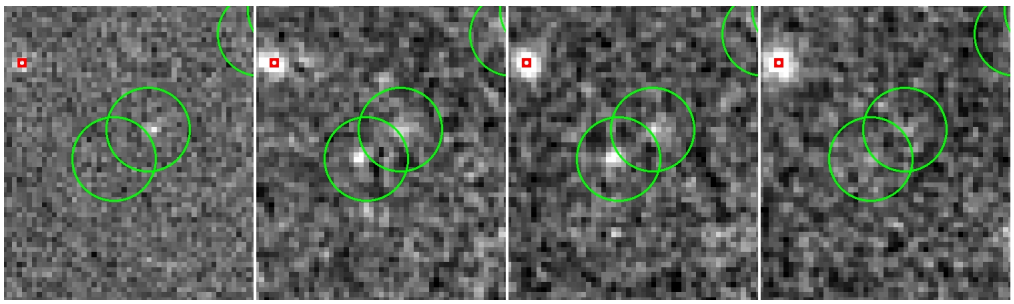


Figura 3.58: Come Fig. 3.57.

Conclusioni

Il problema della decomposizione delle immagini, in particolare allo scopo di rimuoverne il rumore, è tra i più noti e trattati nel campo dei metodi matematici di analisi delle immagini (vedi [11, 15, 8, 14, 13, 16]). Lo scopo di questo lavoro è stato quello di dimostrare come una buona decomposizione "struttura-texture" possa effettivamente essere utile per l'analisi scientifica delle immagini astronomiche, con particolare attenzione all'identificazione di nuove sorgenti comunemente nascoste dal rumore. Sebbene il rumore sia una componente, e un limite, fondamentale nelle immagini astronomiche digitali (Cap.2), non ci risulta siano stati fatti prima di questo lavoro tentativi di applicare algoritmi di denoising a tale genere di immagini. A tale scopo abbiamo preso in considerazione tre diversi metodi di decomposizione basati su tecniche di minimizzazione della variazione totale di un segnale abbinate ad un controllo della norma del segnale stesso (Cap.2). Gli algoritmi sono stati implementati in un codice scritto in C++ chiamato `ATVD` (*Astral Total Variation Denoiser*), opportunatamente ottimizzato e parallelizzato in modo da ottenere le migliori prestazioni possibili sia in esecuzione seriale che in esecuzione parallela su diversi threads, con ottimi risultati di scalabilità (vedi sezione 2.7, in particolare 2.7.5). Abbiamo sperimentato il codice innanzitutto su immagini fotografiche e semplici simulazioni geometriche (sezioni 3.1 e 3.2) mostrando l'efficacia degli algoritmi di decomposizione e del criterio di correlazione in tali contesti. In seguito abbiamo utilizzato gli algoritmi implementati in `ATVD` su simulazioni di immagini astronomiche (sezione 3.3) e su immagini astronomiche reali (sezione 3.4). I test effettuati hanno mostrato che la decomposizione è un approccio estremamente promettente per migliorare l'analisi delle immagini astrofisiche. In particolare, come mostrato nelle sezioni 3.3.5 e 3.4.1, sulle immagini "struttura" può essere individuato

un maggior numero di sorgenti "deboli" di quante possano essere trovate sulle immagini originali. Tramite vari test abbiamo verificato che tali sorgenti sono nella stragrande maggioranza dei casi oggetti reali che prima sfuggivano all'analisi a causa della presenza del rumore.

A seguito dei test svolti abbiamo inoltre individuato delle possibili vie per sviluppi futuri per il codice `ATVD` e lo studio del denosing di immagini astrofisiche. Abbiamo infatti verificato che nonostante l'algoritmo di correlazione riesca ad automatizzare il processo di selezione del lambda migliore, nel caso di simulazioni astronomiche esso non restituisce la migliore decomposizione possibile in termini di `PSNR` come invece fa nel caso di immagini fotografiche "tradizionali". In futuro, test intensivi eseguiti su simulazioni astronomiche con vari livelli di rumore e con minore o maggior numero di sorgenti potranno dirci la causa di questa differenza tra immagini astronomiche e fotografiche e se esista un modo di migliorare l'uso del principio di correlazione nel contesto astrofisico. Un'alternativa possibile potrà essere data dal testare approcci alternativi quali algoritmi basati sulla massimizzazione di stime per il `PNSR` ([26]), algoritmi simili al nostro algoritmo di correlazione opportunatamente modificati per le immagini "multi-scaling" ([25]) oppure approcci diversi dal metodo variazionale come tecniche di denoising che sfruttano la manipolazione dei coefficienti wavelet del segnale ([11]). Abbiamo verificato inoltre che una piccola parte delle sorgenti individuabili sull'immagine struttura non sono reali ma artefatti strumentali o regioni di rumore apparentemente correlato. Ulteriore lavoro sarà necessario per individuare l'utilizzo ottimale dei metodi di decomposizione massimizzandone l'affidabilità e dunque l'utilità per l'analisi fisica delle sorgenti astronomiche. Il nostro obiettivo finale è dunque quello di proporre un programma in grado di decomporre in maniera del tutto automatica e affidabile immagini astronomiche di diverso genere: il lavoro presentato in questa tesi costituisce un primo, promettente passo in questa direzione.

Bibliografia

[1] J.F. Aujol, G. Gilboa, T. Chan, S. Osher. *Structure-Texture image decomposition - modeling, algorithms and parameter selection*, Luglio 2005.

[2] S. Alinei. *A property of the minimum vectors of a regularizing functional defined by means of the absolute norm*, IEEE Transaction on Signal Processing, 45(4):913-917, 1997.

[3] G. Aubert e P.Kornprobst. *Mathematical Promblems in Image Processing*, volume 147 di Applied Mathematical Sciences, Springer-Verlag, 2002.

[4] J.F. Aujol, G. Aubert, L. Blanc-Féraud, A. Chambolle. *Image decomposition into a bounded variation component and an oscillating component*, Journal of Mathematical Imaging and Vision, 22(1):71-88, Gennaio 2005.

[5] J.F. Aujol, G. Aubert, L. Blanc-Féraud, Antonin Chambolle. *Decomposing an image: Application to SAR images. In Scale Space '03*, volume 2695 of Lecture Notes in Computer Science, 2003.

[6] A. Chambolle. *An algorithm for total variation minimization and applications*, JMIV, 20:89-97, 2004.

[7] A. Chambolle e P.L. Lions. *Image recovery via total variation minimization and related problems.* Numerische Mathematik, 76(3):167-188, 1997.

[8] A. Chambolle, R.A. De Vore, N. Lee, e B.J. Lucier. *Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage.* IEEE Transcations on Image Processing, 7(3):319–335, Marzo 1998.

[9] P. L. Combettes and V. R. Wajs. *Theoretical analysis of some regularized image denoising methods*. In ICIP 04, volume 1, pages 969–972, 2004.

[10] P. L. Combettes and V. R. Wajs. *Signal recovery by proximal forward-backward splitting*. SIAM Journal on Multiscale Modeling and Simulation, 2005. in press.

[11] D.L. Donoho and M. Johnstone. *Adapting to unknown smoothness via wavelet shrinkage*. Journal of the American Statistical Association, 90(432):1200–1224, Dicembre 1995.

[12] I. Ekeland and R. Temam. *Analyse convexe et problmes variationnels. Etudes Mathematiques*. Dunod, 1974.

[13] F. Malgouyres. *Mathematical analysis of a model which combines total variation and wavelet for image restoration*. Journal of information processes,2(1):1–10, 2002.

[14] F. Malgouyres. *Minimizing the total variation under a general convex constraint for image restoration*. IEEE transactions on image processing,11(12):1450–1456, Dicembre 2002.

[15] S.G. Mallat. *A theory for multiresolution signal decomposition: The wavelet representation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693, Luglio 1989.

[16] Yves Meyer. *Oscillating patterns in image processing and in some nonlinear evolution equations*, Marzo 2001. The Fifteenth Dean Jacquelines B. Lewis Memorial Lectures.

[17] P. Mrazek and M. Navara. *Selection of optimal stopping time for nonlinear diffusion filtering*. IJCV, 52(2/3):189-203, 2003.

[18] M. Nikolova. *A variational approach to remove outliers and impulse noise*, JMIV, 20(1-2):99-120, 2004.

[19] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. *An iterative regularization method for total variation based image restoration*, 2004. CAM report 04-13, SIAM Journal on Multiscale Modeling and Simulation.

[20] L. Rudin, S. Osher, E. Fatemi. *Nonlinear total variation based noise removal algorithms*, Physica D, 60:259-268, 1992.

[21] L.A. Vese and S.J. Osher. *Modeling textures with total variation minimization and oscillating patterns in image processing.* Journal of Scientific Computing, 19:553–572, 2003.

[22] Zhou Wang, Alan C Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. *Image quality assessment: From error visibility to structural similarity.* Image Processing, IEEE Transactions on, 13(4):600–612, 2004.

[23] Bertin, E. and Arnouts, S. *SExtractor: Software for source extraction.* Provided by the SAO/NASA Astrophysics Data System, Giugno 1996

[24] Grogin, N. A. and Kocevski, D. D. and Faber, S. *CANDELS: The Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey*, Dicembre 2011

[25] Kartic Subr, Cyril Soler *Edge-preserving multiscale image decomposition based on local extrema*, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH, Volume 28 Issue 5, December 2009.

[26] Shashikant Agrawal, Rajkumar Sahu *Wavelet Based MRI Image Denoising Using Thresholding Techniques*, International Journal of Science, Engineering and Technology Research (IJSETR), Settembre 2012.