

2022-2023

# BLS-GSM

FOR IMAGE DENOISING

Roberto Falcone

CORSO: *Elaborazione di immagini*

PROFESSORE: *R. Restaino*

# L'ALGORITMO

Data un'immagine rumorosa in input e considerata una rappresentazione wavelet multi-risoluzione di suddetta immagine, l'idea dell'algoritmo BLS-GSM è di modellare un intorno di un coefficiente della wavelet,  $\mathbf{x}$ , privo di rumore, attraverso una Gaussian scale mixture:

$$\mathbf{x} = \sqrt{z} \mathbf{u}.$$

Dove  $\mathbf{u}$  è un vettore aleatorio gaussiano a media nulla e  $z$  è una variabile aleatoria casuale, indipendente e positiva.

Considerato un vicinato dei coefficienti della wavelet dell'immagine rumorosa, questo può essere espresso come  $\mathbf{y} = \mathbf{x} + \mathbf{w} = \sqrt{z}\mathbf{u} + \mathbf{w}$ , dove  $\mathbf{y}$  è il vicinato rumoroso osservato,  $\mathbf{x}$  è il vicinato originale (privo di rumore) e  $\mathbf{w}$  indica il rumore gaussiano additivo di varianza  $\sigma$ .

L'obiettivo dell'algoritmo è calcolare, sulla base di  $\mathbf{y}$ , una stima BLS di  $\mathbf{x}$  →

$$E\{\mathbf{x}|\mathbf{y}\} = \int_0^{\infty} p(z|\mathbf{y})E(\mathbf{x}|\mathbf{y}, z)dz$$

# IMPLEMENTAZIONE

Ho implementato l'algoritmo come esposto all'interno dell'articolo (Javier Portilla, 2022) fatta eccezione per:

- L'utilizzo della scomposizione in steerable pyramid.
- Decomposizione in autovettori della matrice di covarianza di  $\mathbf{u}$ .

## Funzione

La funzione `BLS_GSM()` implementata prende in input la sola immagine rumorosa e fornisce in output la sola immagine elaborata dall'algoritmo.

Sia l'immagine in input che l'immagine in output sono del tipo `uint8`, tuttavia l'algoritmo, per questioni di performance, lavora sulla conversione a `float64`.

## Padding

Per effettuare la scomposizione dell'immagine a diversi livelli è necessario che le sue dimensioni siano divisibili per  $2^n$ , dove  $n$  sono il numero di livelli nei quali si vuole scomporre l'immagine.

Di conseguenza viene effettuato un primo controllo per verificare la precedente condizione, nel caso essa non fosse soddisfatta si effettua un padding all'immagine (che verrà rimosso una volta finita l'elaborazione).

## Stima del rumore

Dato che l'algoritmo si basa sulla suddivisione di un profilo di rumore è stato necessario costruirne uno a partire da una stima di quello presente all'interno dell'immagine.

La stima è stata effettuata applicando un box filter all'immagine rumorosa e sottraendo la varianza dell'immagine filtrata a quella dell'immagine rumorosa.

Nel caso di immagini a colori la stima è stata effettuata per ognuno dei tre canali.

## Parametri

Alcuni parametri, come i livelli di decomposizione wavelet, dipendono strettamente dalle dimensioni dell'immagine. Al contrario, altri parametri, come lo step per la discretizzazione dell'integrale su  $z$ , restano sempre fissati.

Per osservare nel dettaglio la lista dei parametri e dei valori scelti consultare l'appendice A.

# TESTS ON SYNTHETIC NOISY IMAGES

Le performance dell'algoritmo sono state valutate su un test set di immagini. La prima valutazione di performance è stata effettuata con tutte immagini di  $512 \times 512$  pixel, alle quali è stato aggiunto un rumore gaussiano con varianza crescente. Infine, allo scopo di valutare le performance dell'algoritmo in casi particolari, sono stati effettuati test su diversi tipi di rumore per verificare la sensibilità dell'algoritmo ad essi.

**NB:** Siccome sono presenti problemi di bordo non risolti, il confronto tra le immagini g.t. e le immagini risultanti dall'algoritmo è stato effettuato ignorando i bordi dove si presenta il problema.

## TEST SET



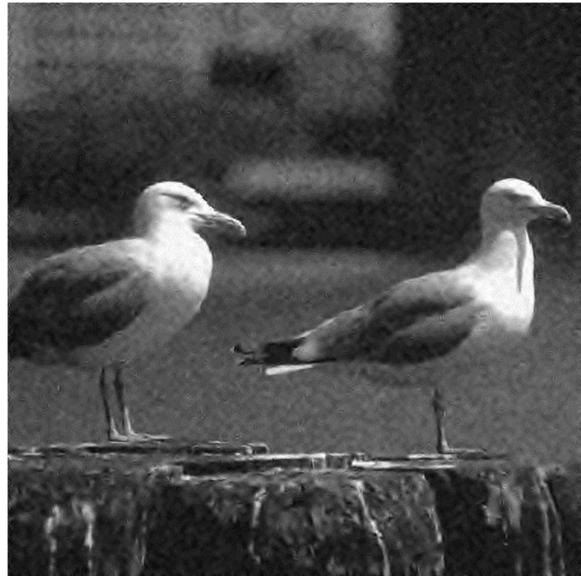
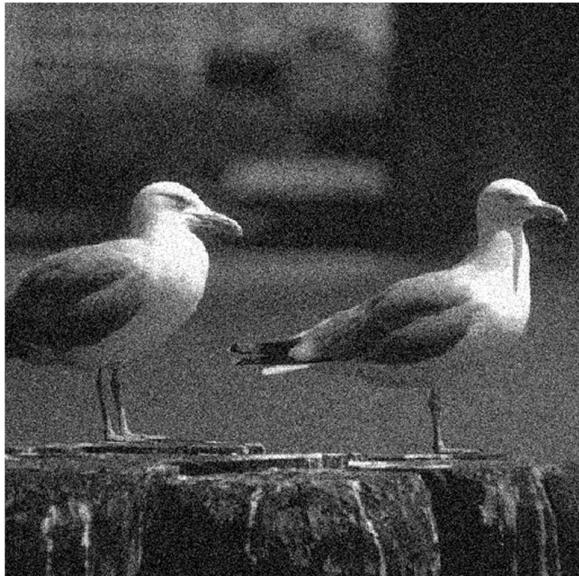
1    2    3    4

Numero immagine	Nome immagine
1	<i>Birds</i>
2	<i>Boat</i>
3	<i>Lenna</i>
4	<i>Parrots</i>

## I test (Varianza del rumore)

A sinistra possiamo osservare le immagini rumorose, a destra le immagini elaborate dall'algoritmo

$$var = 0.1$$



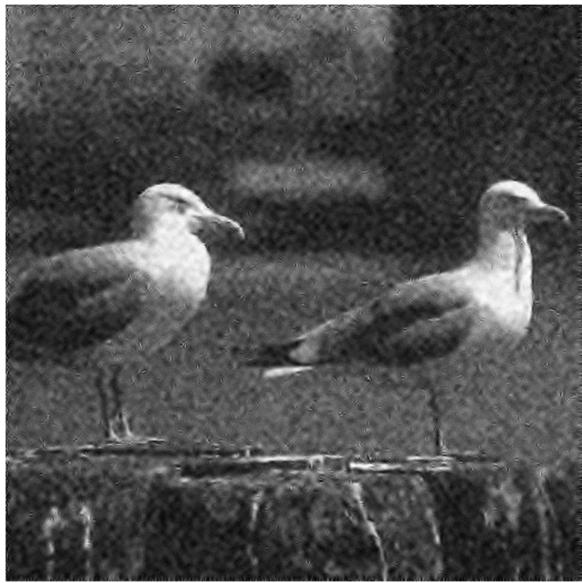


### Performances

Sono state valutate le performance analizzando i principali indici di qualità (Mean squared error, Peak signal-to-noise ratio, Structural similarity)

<i>Immagine</i>	<i>MSE</i>	<i>PSNR</i>	<i>SS</i>
<b>1</b>	64.088	30.063	0.752
<b>2</b>	89.587	28.608	0.758
<b>3</b>	73.588	29.462	0.739
<b>4</b>	62.209	30.192	0.769

$var = 0.2$





### Performances

<i>Immagine</i>	<i>MSE</i>	<i>PSNR</i>	<i>SS</i>
<i>Birds</i>	187.900	25.391	0.547
<i>Boat</i>	217.653	24.753	0.574
<i>Lenna</i>	194.273	25.246	0.561
<i>Parrots</i>	191.716	25.304	0.560

## Confronto dei risultati con altri metodi di denoising noti

Prendiamo in considerazione il caso con  $var = 0.1$

### MSE

Immagine	BLS-GSM	Bilateral	Box filter	Median	Wiener
Birds	<u>64.088</u>	145.216	99.882	134.301	303.755
Boat	<u>89.587</u>	159.275	124.94	161.893	753.517
Lenna	<u>73.588</u>	219.382	106.10	143.262	1842.203
Parrots	<u>62.209</u>	212.394	115.83	136.099	916.456

### PSNR

Immagine	BLS-GSM	Bilateral	Box filter	Median	Wiener
Birds	<u>30.063</u>	26.510	28.13	26.849	23.305
Boat	<u>28.608</u>	26.109	27.16	26.038	19.359
Lenna	<u>29.462</u>	24.718	27.87	26.569	15.477
Parrots	<u>30.192</u>	24.859	27.49	26.792	18.509

### SS

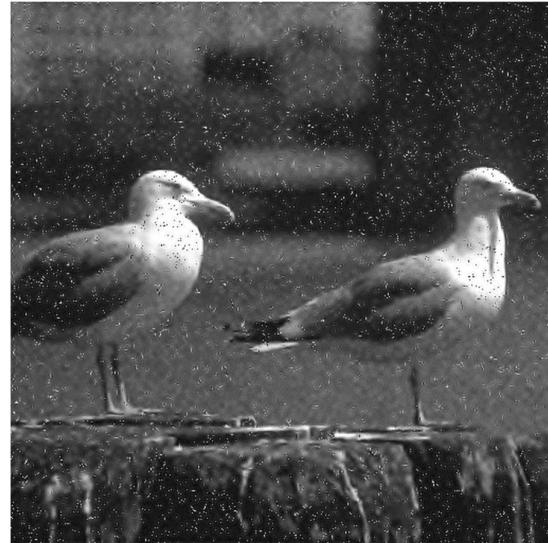
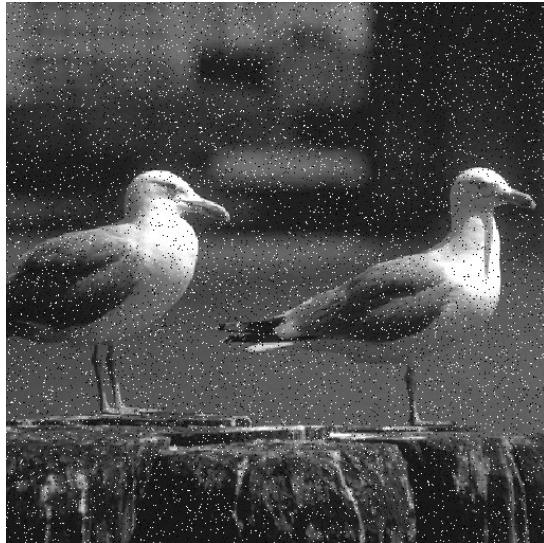
Immagine	BLS-GSM	Bilateral	Box filter	Median	Wiener
Birds	<u>0.751</u>	0.492	0.609	0.520	0.579
Boat	<u>0.758</u>	0.575	0.650	0.580	0.571
Lenna	<u>0.739</u>	0.471	0.625	0.548	0.576
Parrots	<u>0.769</u>	0.423	0.624	0.541	0.654

Per il confronto degli indici di qualità in tutti gli altri test effettuati consultare l'appendice A.

## Altri test

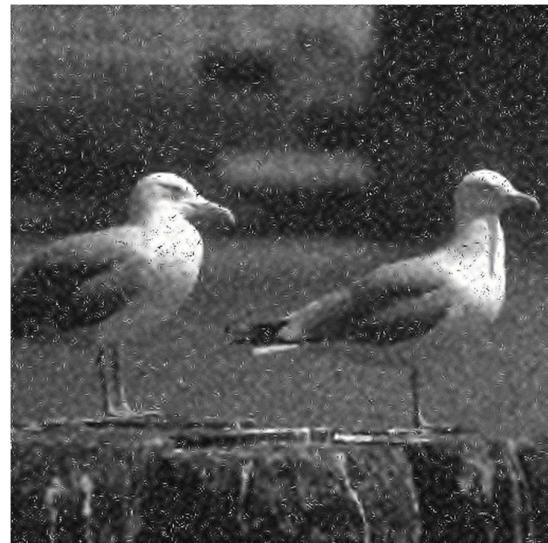
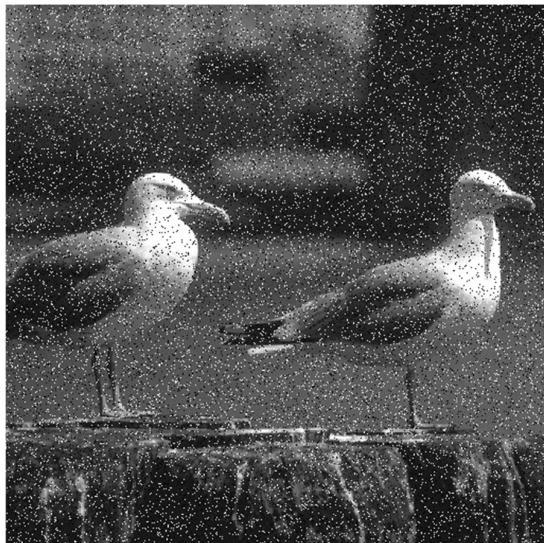
Rumore sale e pepe

Intensità → 0.05



Metric	BLS-GSM	Bilateral	Box filter	Median	Wiener
MSE	404.061	760.330	159.883	<u>23.277</u>	642.782
PSNR	22.066	19.320	26.092	<u>34.461</u>	20.050
SS	0.499	0.381	0.547	<u>0.924</u>	0.387

Intensità → 0.1

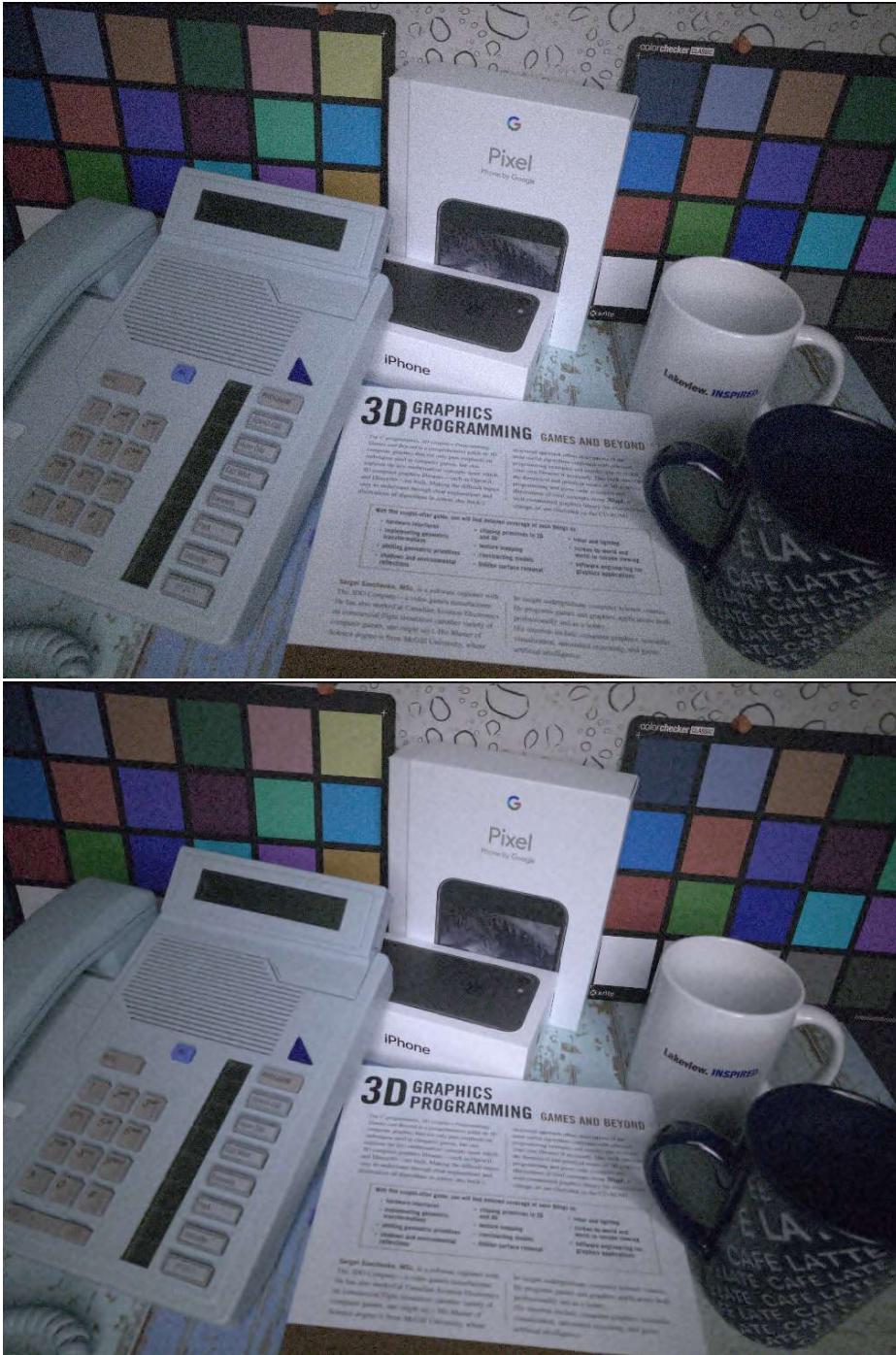


Metric	BLS-GSM	Bilateral	Box filter	Median	Wiener
MSE	451.236	1542.083	308.249	<u>28.917</u>	930.111
PSNR	21.586	16.249	23.241	<u>33.519</u>	18.445
SS	0.445	0.197	0.395	<u>0.919</u>	0.249

# TESTS ON REAL NOISY IMAGES

Le performance dell'algoritmo sono state successivamente valutate anche su due immagini che presentano già un rumore gaussiano.

## I Immagine



## II Immagine

Data la grandezza della seconda immagine, è stato deciso di ridurne le dimensioni del 75% per avvicinarsi a quelle della prima immagine reale.



Performances prima immagine

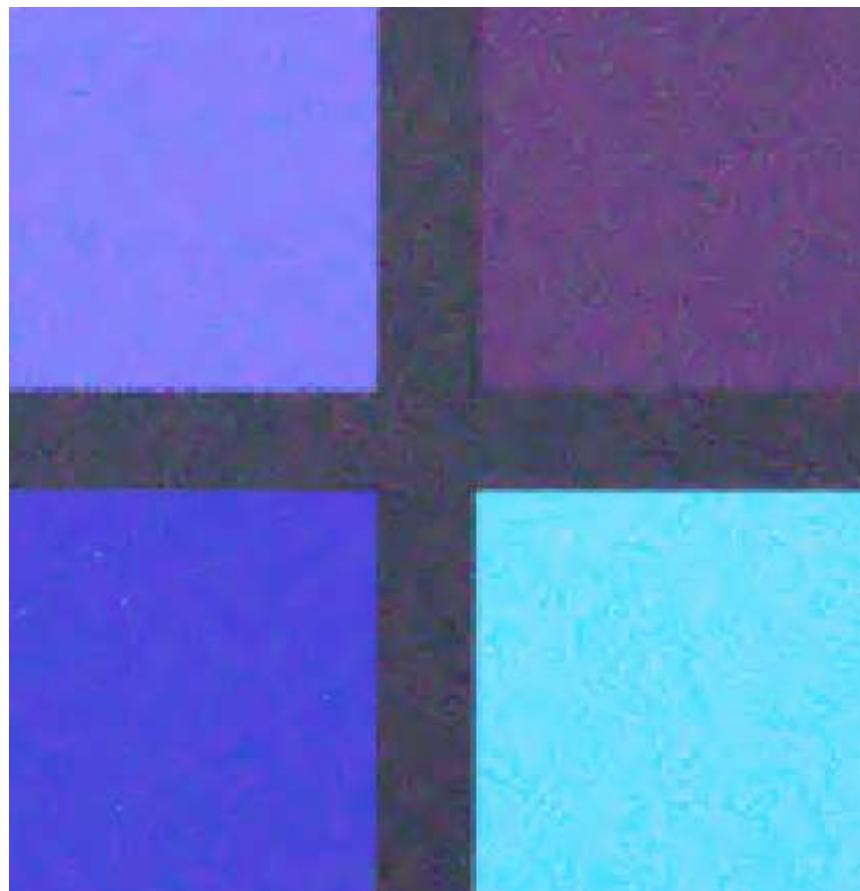
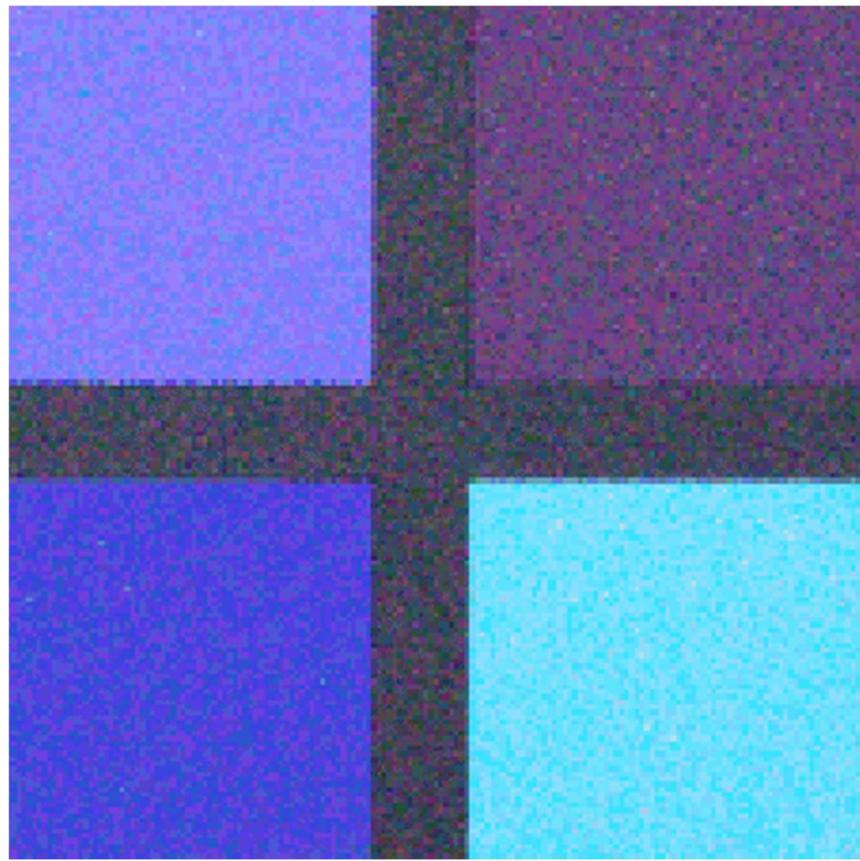
Metric	BLS-GSM	Bilateral	Box filter	Median	Wiener
MSE	56.702	73.899	<u>53.284</u>	60.319	479.095
PSNR	30.594	29.444	<u>30.864</u>	30.326	21.326
SS	<u>0.804</u>	0.70	0.794	0.743	0.745

Performances seconda immagine

Metric	BLS-GSM	Bilateral	Box filter	Median	Wiener
MSE	<u>146.117</u>	182.060	150.164	165.302	1370.126
PSNR	<u>26.483</u>	25.528	26.365	25.948	16.763
SS	0.673	0.631	<u>0.736</u>	0.665	<u>0.736</u>

## Confronto nel dettaglio





# CONSIDERAZIONI FINALI

## Performances

Analizzando i test effettuati sulle immagini con rumore aggiunto possiamo notare come le performance dell'algoritmo siano al quanto ottime, in quanto nessun altro metodo, in nessuna delle metriche, supera il risultato ottenuto da BLS-GSM.

Tuttavia, nel caso delle immagini reali, le differenze tra i vari metodi si assottigliano, ma le performance dell'algoritmo sviluppato rimangono decisamente buone, restando comunque molto vicine ai risultati migliori, ossia, quelli ottenuti attraverso un filtraggio con box-filter.

Nel caso di rumore del tipo sale e pepe, il miglior metodo risulta essere il filtro mediano, in quanto il meno sensibile a valori "molto fuori norma". Tuttavia, in entrambi i casi analizzati per questo tipo di rumore, il metodo BLS-GSM si classifica terzo, con prestazioni paragonabili a quelle del Box Filter (secondo).

## Complessità

La complessità temporale dell'algoritmo dipende dalla quantità di pixel dell'immagine. Siano  $m$  e  $n$  altezza e larghezza dell'immagine, la complessità dell'algoritmo è nell'ordine di  $O(m \cdot n)$ .

## Tempo di esecuzione

Dai test effettuati l'algoritmo impiega circa 8 minuti per elaborare immagini monocromatiche  $512x512$  pixels. Nel caso di immagini a colori della stessa dimensione invece il tempo di esecuzione oscilla intorno ai 30 minuti. Circa 1 ora invece è stata necessaria per elaborare le immagini reali  $1300x700$  pixels circa.

# APPENDICE A

## Liste dei parametri

$z_{step}$	2
$z_{min}$	$1.25 \cdot 10^{-9}$
$z_{max}$	33.1155
$wavelet$	<i>sym3</i>
$wavelet_{levels}$	$\frac{\log_2(\min\{N, M\} - 4)}{2}$
$neighboorhood_{size}$	(3x3)

## Altri test e immagini salvate

All'interno dell'archivio "Test.rar" consegnato sono presenti i log, assieme alle immagini in formato bmp, di tutti i test analizzati all'interno di questo report, al contempo sono presenti anche il test set utilizzato e i risultati di altri test non approfonditi. L'organizzazione delle cartelle e dei file di log è indicata all'interno del file "README.txt".