

Relazione Progetto di Calcolo Numerico

"Deblur Immagini"

Benatti Alice, Manuelli Matteo, Qayyum Shahbaz Ali

Gennaio 2022

"Di sicuro, ci sarà sempre chi guarderà solo la tecnica e si chiederà "come", mentre altri di natura più curiosa si chiederanno "perché"."

— *Man Ray*
(fotografo statunitense)



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

Indice

1 Presentazione del problema	3
1.1 Generazione dataset	3
1.2 Generazione Immagini Corrotte	4
1.3 Osservazioni	10
1.3.1 Analisi immagine geometrica	10
1.3.2 Analisi immagini fotografiche	10
2 Ricostruzione di un immagine rispetto una versione corrotta	12
2.1 Metodo del Gradiente Coniugato (naive)	12
2.2 Metodo del Gradiente	13
2.3 Metodo del Gradiente e Metodo del Gradiente Coniugato a confronto	15
3 Metodi di Regolarizzazione	17
3.1 Metodo di Regolarizzazione con Metodo del Gradiente <code>minimize</code>	17
3.1.1 Immagini geometriche regolarizzate	17
3.1.2 Immagini fotografiche regolarizzate	19
3.2 Metodo di Regolarizzazione di Tikhonov	20
4 Variazione Totale	23
4.1 Variazione totale dell'immagine fotografica pugile:	24
4.2 Variazione totale dell'immagine con testo:	25
5 Conclusioni	25

1 Presentazione del problema

Il progetto ha come scopo quello di comprendere e mettere in atto metodi per ricostruire immagini blurate e svolgere il lavoro opposto, quindi generare immagini corrotte dal rumore a partire da un'immagine originale.

Il problema che ci è stato presentato riguarda la ricostruzione di immagini corrotte attraverso il blur Gaussiano.

Verrà analizzata inizialmente l'immagine `data.camera()` importata da `skimage`, successivamente verranno analizzate un set di 8 immagini con oggetti geometrici di colore uniforme su sfondo nero, realizzate da noi.

Il problema di deblur consiste nella ricostruzione di un'immagine a partire da un dato acquisito mediante il seguente modello:

$$b = Ax + \eta$$

dove b rappresenta l'immagine corrotta, x l'immagine originale che vogliamo ricostruire, A l'operatore che applica il blur Gaussiano ed η il rumore additivo con distribuzione Gaussiana di media μ e deviazione standard σ .

Per svolgere il progetto si farà uso dei moduli `numpy`, `skimage` e `matplotlib` utilizzando il linguaggio Python.

Affinché risultino chiari i valori a cui andremo a riferirci nella relazione, bisogna tenere ben presente il significato di questi due parametri.

PSNR (Peak Signal to Noise Ratio): Misura la qualità di un'immagine ricostruita rispetto all'immagine originale, la formula per calcolarlo è la seguente:

$$PSNR = \log_{10}\left(\frac{\max x^*}{\sqrt{MSE}}\right)$$

MSE (Mean Squared Error): Con la sigla ci riferiamo all'errore quadratico medio ed è così ottenuto:

$$MSE = \sqrt{\frac{\sum_{i=1}^n \sum_{j=1}^m (x_{ij}^* - x_{ij})^2}{nm}}$$

I due valori sono inversamente proposizionali, quindi più è alto il PSNR e basso l'MSE, più l'immagine sarà simile all'immagine originale. Il PSNR dipende dall'MSE.

Deviazione standard: E' un indice che ci permette di capire in maniera riassuntiva le differenze dei valori per ogni osservazioni rispetto alla media delle variabili.

1.1 Generazione dataset

E' richiesto un set di immagini con le seguenti specifiche:

- 8 Immagini di dimensione 512×512 ;
- Formato PNG in scala di grigi;
- Devono contenere tra i 2 ed i 6 oggetti geometrici;
- Oggetti di colore uniforme su uno sfondo nero.

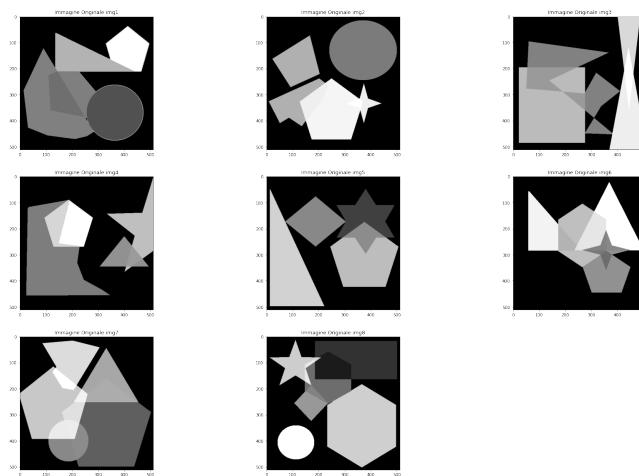


Figura 1: Immagini geometriche utilizzate

Useremo anche altre due immagini di tipo fotografico/medico/astronomico a scelta trovate su internet. Quest'ultime saranno importate all'interno del progetto con la libreria `skimage`, impostando il flag `as_gray=True` per averle in bianco e nero.

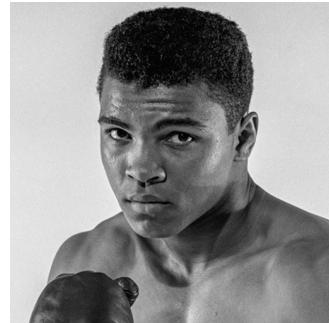
Le immagini selezionate sono le seguenti:

Immagine Fotografica Mostra un fotografo nell'intento di uno scatto con sfondo paesaggistico.

Immagine Ritratto Ritrae il volto di una persona in modo dettagliato e con varie tonalità di grigio.



(a) Immagine fotografica



(b) Immagine ritratto

Figura 2: Immagini fotografiche analizzate

1.2 Generazione Immagini Corrotte

Obiettivo: Degradoare le immagini applicando, mediante le funzioni riportate nella cella precedente, l'operatore di blur con parametri

- $\sigma = 0.5$ dimensione 5×5
- $\sigma = 1$ dimensione 7×7
- $\sigma = 1.3$ dimensione 9×9

ed aggiungere rumore gaussiano con diversi valori di deviazione standard.

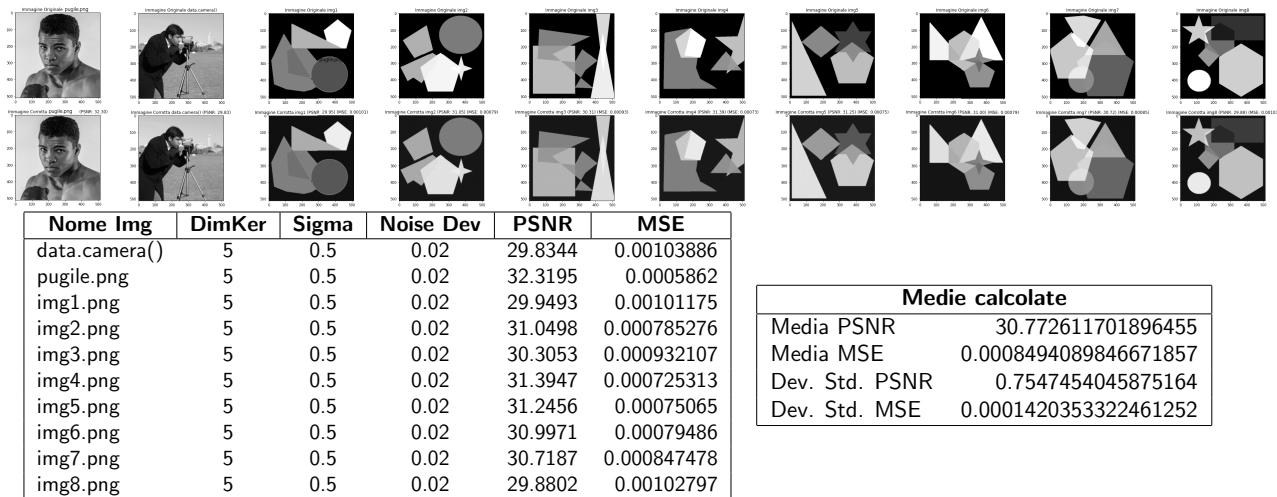
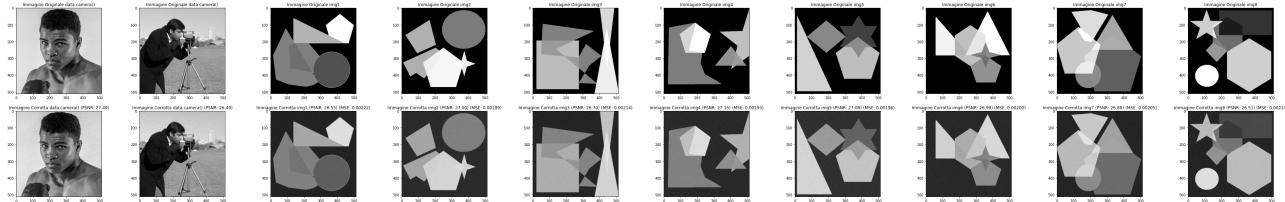


Figura 3 & Tabella 1: Immagini corrotte con $\sigma = 0.5$ dimensione 5×5 e noise = 0.02

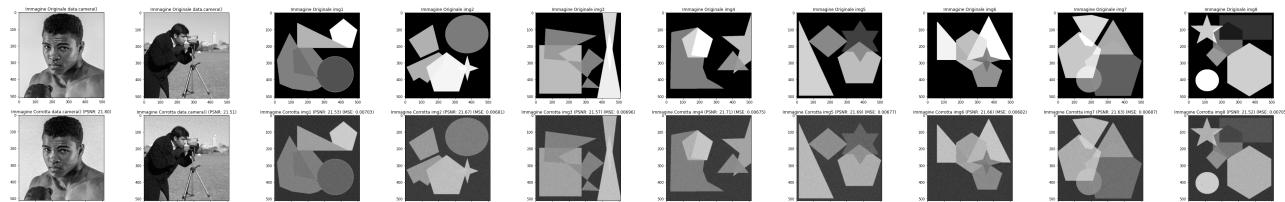


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	5	0.5	0.04	26.5112	0.00223297
pugile.png	5	0.5	0.04	27.4667	0.0017919
img1.png	5	0.5	0.04	26.5452	0.00221553
img2.png	5	0.5	0.04	27.0007	0.00199492
img3.png	5	0.5	0.04	26.6982	0.00213887
img4.png	5	0.5	0.04	27.1549	0.00192537
img5.png	5	0.5	0.04	27.0846	0.00195677
img6.png	5	0.5	0.04	26.9876	0.00200098
img7.png	5	0.5	0.04	26.8806	0.00205086
img8.png	5	0.5	0.04	26.5063	0.00223545

Medie calcolate

Media PSNR	26.891734569652385
Media MSE	0.002050504949458885
Dev. Std. PSNR	0.3007954092260605
Dev. Std. MSE	0.00014057036112999103

Figura 4 & Tabella 2: Immagini corrotte con $\sigma = 0.5$ dimensione 5×5 e noise = 0.04

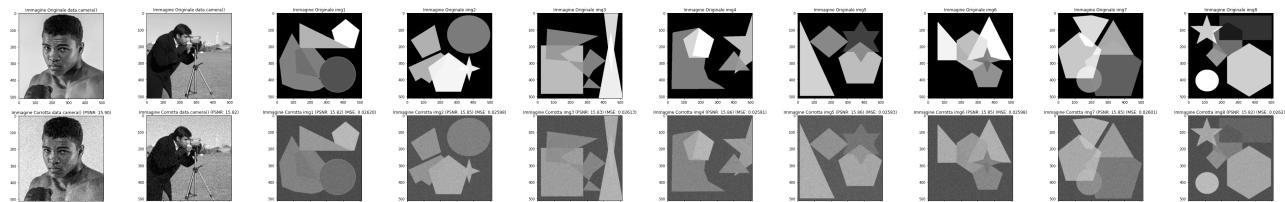


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	5	0.5	0.08	21.5325	0.00702666
pugile.png	5	0.5	0.08	21.7986	0.0066091
img1.png	5	0.5	0.08	21.5332	0.00702558
img2.png	5	0.5	0.08	21.6702	0.00680731
img3.png	5	0.5	0.08	21.5736	0.00696048
img4.png	5	0.5	0.08	21.7089	0.00674704
img5.png	5	0.5	0.08	21.6929	0.00677191
img6.png	5	0.5	0.08	21.6626	0.00681931
img7.png	5	0.5	0.08	21.633	0.00686601
img8.png	5	0.5	0.08	21.519	0.00704858

Medie calcolate

Media PSNR	21.656429313389726
Media MSE	0.00683046240248098
Dev. Std. PSNR	0.08995967536472305
Dev. Std. MSE	0.00014118437500020744

Figura 5 & Tabella 3: Immagini corrotte con $\sigma = 0.5$ dimensione 5×5 e noise = 0.08

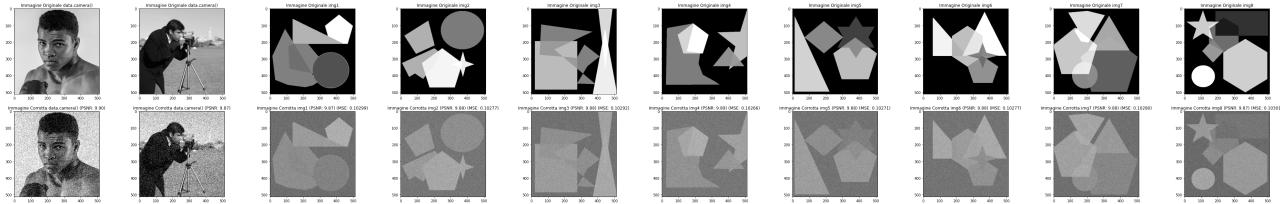


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	5	0.5	0.16	15.7869	0.0263819
pugile.png	5	0.5	0.16	15.89	0.0257632
img1.png	5	0.5	0.16	15.8162	0.0262048
img2.png	5	0.5	0.16	15.8543	0.025976
img3.png	5	0.5	0.16	15.8278	0.0261347
img4.png	5	0.5	0.16	15.8649	0.0259124
img5.png	5	0.5	0.16	15.8618	0.0259313
img6.png	5	0.5	0.16	15.8537	0.0259797
img7.png	5	0.5	0.16	15.849	0.0260077
img8.png	5	0.5	0.16	15.8161	0.0262051

Medie calcolate

Media PSNR	15.821291629768288
Media MSE	0.026174436676272776
Dev. Std. PSNR	0.02377710929356369
Dev. Std. MSE	0.0001432278885193189

Figura 6 & Tabella 4: Immagini corrotte con $\sigma = 0.5$ dimensione 5×5 e noise = 0.16

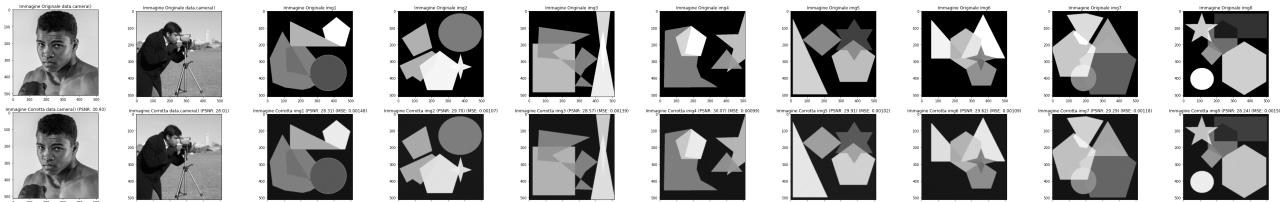


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	5	0.5	0.32	9.90122	0.102301
pugile.png	5	0.5	0.32	9.89115	0.102538
img1.png	5	0.5	0.32	9.8721	0.102989
img2.png	5	0.5	0.32	9.88127	0.102772
img3.png	5	0.5	0.32	9.87516	0.102916
img4.png	5	0.5	0.32	9.8862	0.102655
img5.png	5	0.5	0.32	9.88405	0.102706
img6.png	5	0.5	0.32	9.88154	0.102765
img7.png	5	0.5	0.32	9.88	0.102802
img8.png	5	0.5	0.32	9.87102	0.103014

Medie calcolate

Media PSNR	9.908921219921675
Media MSE	0.10211940428183572
Dev. Std. PSNR	0.005854966950232008
Dev. Std. MSE	0.00013766495799610393

Figura 7 & Tabella 5: Immagini corrotte con $\sigma = 0.5$ dimensione 5×5 e noise = 0.32

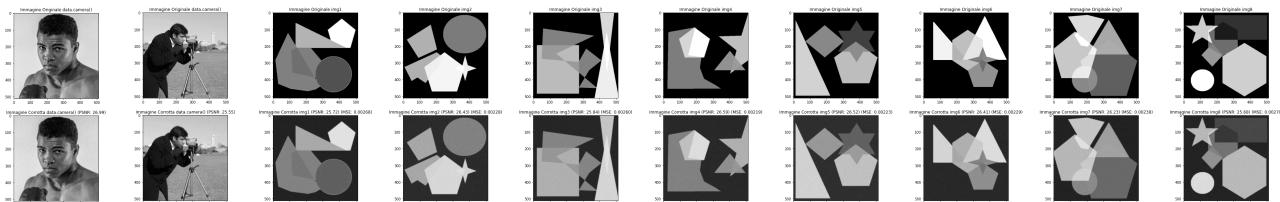


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	7	1	0.02	27.9965	0.00158617
pugile.png	7	1	0.02	30.9284	0.000807527
img1.png	7	1	0.02	28.3054	0.00147726
img2.png	7	1	0.02	29.7038	0.00107057
img3.png	7	1	0.02	28.5689	0.00139029
img4.png	7	1	0.02	30.0656	0.000985011
img5.png	7	1	0.02	29.9064	0.0010218
img6.png	7	1	0.02	29.6228	0.00109074
img7.png	7	1	0.02	29.2918	0.00117711
img8.png	7	1	0.02	28.2434	0.00149851

Medie calcolate

Media PSNR	29.2560497491376
Media MSE	0.0012124124340562978
Dev. Std. PSNR	0.903380822617657
Dev. Std. MSE	0.00024716496411134373

Figura 8 & Tabella 6: Immagini corrotte con $\sigma = 1$ dimensione 7×7 e noise = 0.02

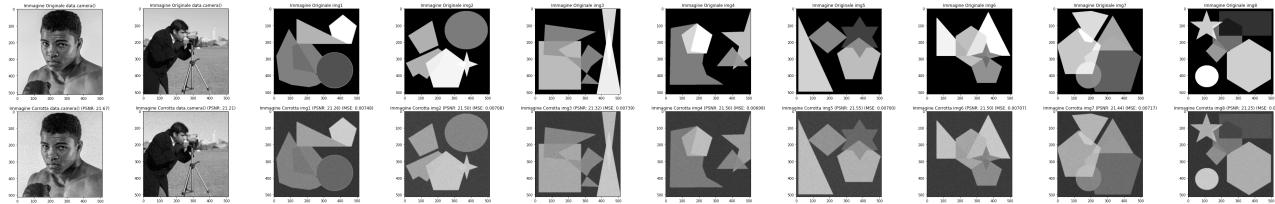


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	7	1	0.04	25.5438	0.00279009
pugile.png	7	1	0.04	26.9696	0.00200928
img1.png	7	1	0.04	25.7196	0.00267942
img2.png	7	1	0.04	26.4282	0.00227604
img3.png	7	1	0.04	25.8439	0.00260379
img4.png	7	1	0.04	26.5898	0.0021929
img5.png	7	1	0.04	26.5209	0.00222797
img6.png	7	1	0.04	26.4067	0.00228732
img7.png	7	1	0.04	26.2277	0.00238357
img8.png	7	1	0.04	25.6828	0.00270224

Medie calcolate

Media PSNR	26.195844232338864
Media MSE	0.0024139860627763257
Dev. Std. PSNR	0.4492018368928465
Dev. Std. MSE	0.0002491578891141964

Figura 9 & Tabella 7: Immagini corrotte con $\sigma = 1$ dimensione 7×7 e noise = 0.04

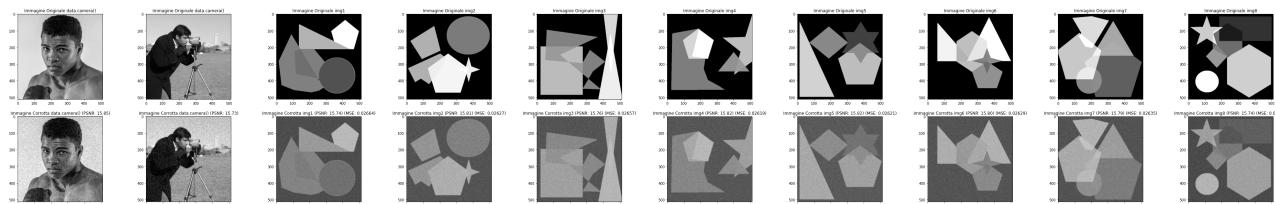


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	7	1	0.08	21.2232	0.0075453
pugile.png	7	1	0.08	21.6733	0.00680245
img1.png	7	1	0.08	21.2586	0.00748406
img2.png	7	1	0.08	21.5025	0.00707543
img3.png	7	1	0.08	21.3156	0.0073866
img4.png	7	1	0.08	21.5582	0.00698523
img5.png	7	1	0.08	21.5463	0.00700433
img6.png	7	1	0.08	21.5043	0.00707252
img7.png	7	1	0.08	21.4429	0.00717307
img8.png	7	1	0.08	21.2464	0.00750518

Medie calcolate

Media PSNR 21.429621292884747
 Media MSE 0.0071992640601343085
 Dev. Std. PSNR 0.14744138849178753
 Dev. Std. MSE 0.0002443541830272355

Figura 10 & Tabella 8: Immagini corrotte con $\sigma = 1$ dimensione 7×7 e noise = 0.08

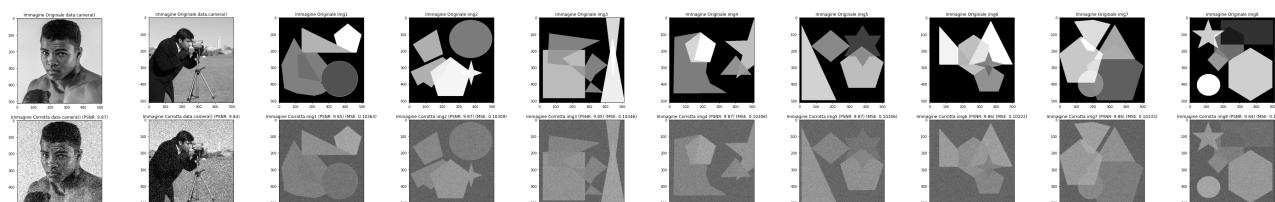


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	7	1	0.16	15.7309	0.0267244
pugile.png	7	1	0.16	15.8496	0.0260039
img1.png	7	1	0.16	15.7444	0.0266415
img2.png	7	1	0.16	15.8054	0.0262698
img3.png	7	1	0.16	15.7565	0.0265673
img4.png	7	1	0.16	15.8194	0.0261856
img5.png	7	1	0.16	15.8151	0.0262111
img6.png	7	1	0.16	15.8029	0.0262853
img7.png	7	1	0.16	15.7926	0.0263473
img8.png	7	1	0.16	15.7385	0.0266776

Medie calcolate

Media PSNR 15.79693992935469
 Media MSE 0.026322419224025406
 Dev. Std. PSNR 0.04145677255353015
 Dev. Std. MSE 0.00025141610640160646

Figura 11 & Tabella 9: Immagini corrotte con $\sigma = 1$ dimensione 7×7 e noise = 0.16

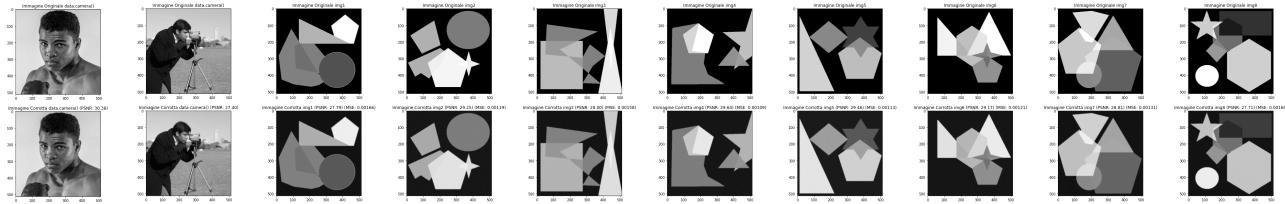


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	7	1	0.32	9.85731	0.10334
pugile.png	7	1	0.32	9.87496	0.102921
img1.png	7	1	0.32	9.84505	0.103632
img2.png	7	1	0.32	9.86765	0.103094
img3.png	7	1	0.32	9.85214	0.103463
img4.png	7	1	0.32	9.86913	0.103059
img5.png	7	1	0.32	9.869	0.103062
img6.png	7	1	0.32	9.86252	0.103216
img7.png	7	1	0.32	9.85775	0.10333
img8.png	7	1	0.32	9.84472	0.10364

Medie calcolate

Media PSNR 9.868096562243034
 Media MSE 0.10308404121278847
 Dev. Std. PSNR 0.009739789869654458
 Dev. Std. MSE 0.00023117252120505893

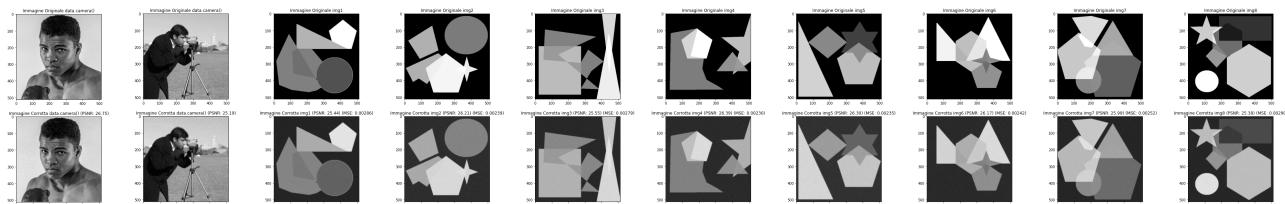
Figura 12 & Tabella 10: Immagini corrotte con $\sigma = 1$ dimensione 7×7 e noise = 0.32



Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	9	1.3	0.02	27.4138	0.00181392
pugile.png	9	1.3	0.02	30.3576	0.0009209
img1.png	9	1.3	0.02	27.7886	0.00166394
img2.png	9	1.3	0.02	29.2539	0.00118745
img3.png	9	1.3	0.02	28.0014	0.00158437
img4.png	9	1.3	0.02	29.6382	0.00108688
img5.png	9	1.3	0.02	29.4566	0.00113329
img6.png	9	1.3	0.02	29.1727	0.00120986
img7.png	9	1.3	0.02	28.8132	0.00131424
img8.png	9	1.3	0.02	27.7142	0.00169269

Medie calcolate	
Media PSNR	28.75660830135966
Media MSE	0.0013622992560759118
Dev. Std. PSNR	0.9328668712951993
Dev. Std. MSE	0.0002893379298389771

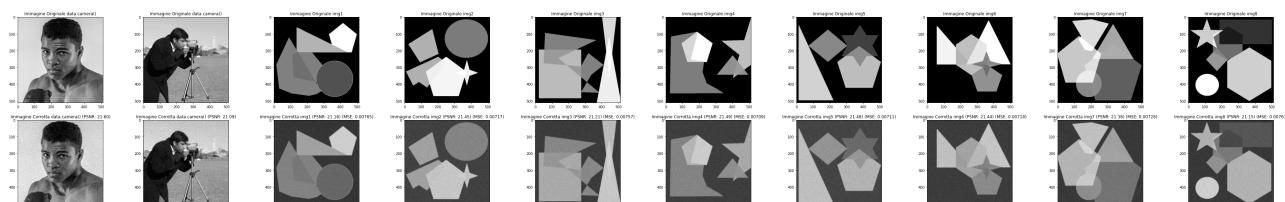
Figura 13 & Tabella 11: Immagini corrotte con $\sigma = 1.3$ dimensione 9×9 e noise = 0.02



Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	9	1.3	0.04	25.1912	0.00302606
pugile.png	9	1.3	0.04	26.7366	0.00212
img1.png	9	1.3	0.04	25.4375	0.00285925
img2.png	9	1.3	0.04	26.2124	0.00239202
img3.png	9	1.3	0.04	25.5474	0.0027878
img4.png	9	1.3	0.04	26.389	0.00229666
img5.png	9	1.3	0.04	26.2972	0.00234571
img6.png	9	1.3	0.04	26.1696	0.00241568
img7.png	9	1.3	0.04	25.9907	0.00251728
img8.png	9	1.3	0.04	25.3782	0.00289853

Medie calcolate	
Media PSNR	25.935031436703177
Media MSE	0.0025660139420201973
Dev. Std. PSNR	0.4896546967680673
Dev. Std. MSE	0.000289707809390377

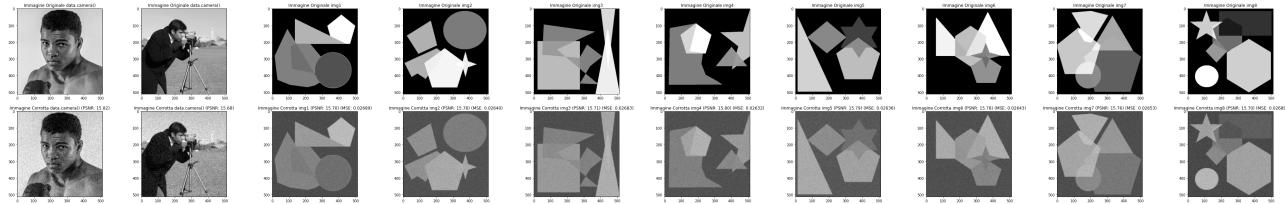
Figura 14 & Tabella 12: Immagini corrotte con $\sigma = 1.3$ dimensione 9×9 e noise = 0.04



Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	9	1.3	0.08	21.0692	0.00781774
pugile.png	9	1.3	0.08	21.6072	0.0069068
img1.png	9	1.3	0.08	21.1615	0.0076534
img2.png	9	1.3	0.08	21.446	0.00716802
img3.png	9	1.3	0.08	21.2099	0.00756858
img4.png	9	1.3	0.08	21.4931	0.00709073
img5.png	9	1.3	0.08	21.4791	0.0071136
img6.png	9	1.3	0.08	21.436	0.0071845
img7.png	9	1.3	0.08	21.3758	0.00728476
img8.png	9	1.3	0.08	21.1529	0.00766845

Medie calcolate	
Media PSNR	21.32346241378315
Media MSE	0.007378763783377743
Dev. Std. PSNR	0.169144238763136
Dev. Std. MSE	0.0002879965860764816

Figura 15 & Tabella 13: Immagini corrotte con $\sigma = 1.3$ dimensione 9×9 e noise = 0.08

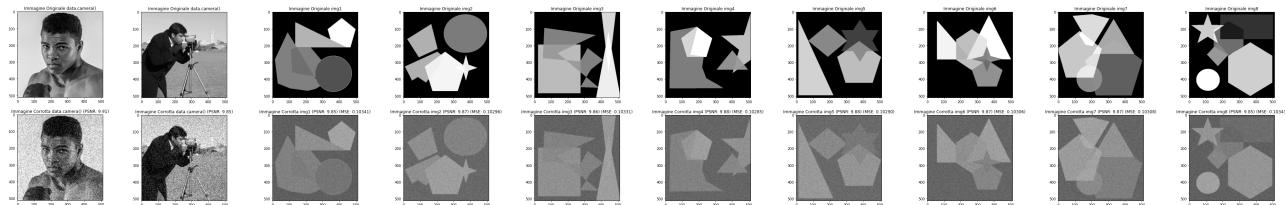


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	9	1.3	0.16	15.6736	0.0270796
pugile.png	9	1.3	0.16	15.8334	0.0261013
img1.png	9	1.3	0.16	15.7044	0.026888
img2.png	9	1.3	0.16	15.7843	0.0263979
img3.png	9	1.3	0.16	15.7146	0.0268251
img4.png	9	1.3	0.16	15.7966	0.026323
img5.png	9	1.3	0.16	15.7911	0.0263568
img6.png	9	1.3	0.16	15.7794	0.0264277
img7.png	9	1.3	0.16	15.7632	0.0265265
img8.png	9	1.3	0.16	15.7044	0.0268884

Medie calcolate

Media PSNR	15.752586448516155
Media MSE	0.026593095526593946
Dev. Std. PSNR	0.04888788575026526
Dev. Std. MSE	0.00029971901539735954

Figura 16 & Tabella 14: Immagini corrotte con $\sigma = 1.3$ dimensione 9×9 e noise = 0.16

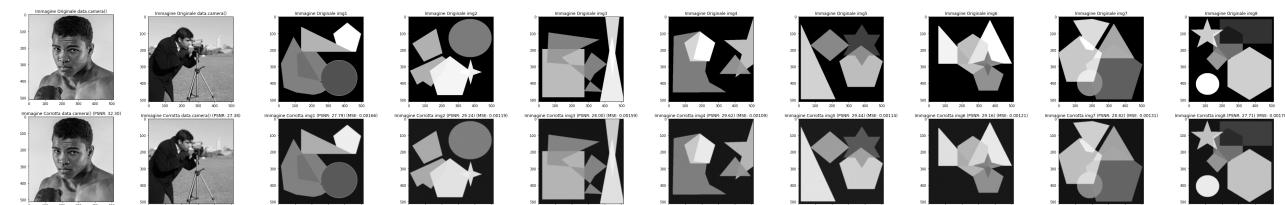


Nome Img	DimKer	Sigma	Noise Dev	PSNR	MSE
data.camera()	9	1.3	0.32	9.82272	0.104167
pugile.png	9	1.3	0.32	9.88385	0.10271
img1.png	9	1.3	0.32	9.85454	0.103406
img2.png	9	1.3	0.32	9.87346	0.102957
img3.png	9	1.3	0.32	9.85875	0.103306
img4.png	9	1.3	0.32	9.87778	0.102854
img5.png	9	1.3	0.32	9.87599	0.102897
img6.png	9	1.3	0.32	9.86911	0.10306
img7.png	9	1.3	0.32	9.8681	0.103084
img8.png	9	1.3	0.32	9.85336	0.103434

Medie calcolate

Media PSNR	9.842576581447004
Media MSE	0.10369173965451903
Dev. Std. PSNR	0.012567871393636715
Dev. Std. MSE	0.0003001770981803593

Figura 17 & Tabella 15: Immagini corrotte con $\sigma = 1.3$ dimensione 9×9 e noise = 0.32



Istanza del Problema	
Media PSNR	28.711752431711446
Media MSE	0.00136419069257633
Dev. Std. PSNR	0.7204441561553295
Dev. Std. MSE	0.00023032931081567697

Figura 18 & Tabella 16: Immagini corrotte con noise = 0.02

Notiamo che tutte le immagini all'aumentare del valore della deviazione standard peggiorano qualitativamente, con $\sigma = 0.5$ e deviazione standard 0.02 abbiamo per esempio un PSNR in media vicino al 30. Invece aumentando la deviazione standard a 0.32 scende addirittura attorno al 9 con delle immagini che tendono a risultare molto sfocate e 'disturbate' dal rumore guassiano. All'aumentare delle dimensioni di Kernel a $\sigma = 1$ di dimensioni 7×7 notiamo che partendo da una dev. standard 0.02 abbiamo un PSNR in media leggermente inferiore rispetto alla precedente dimensione del Kernel. La cosa si ripete quando si passa a $\sigma = 1.3$: i valori di PSNR delle rispettive immagini decrescono rispetto ai valori ottenuti dalle precedenti dimensioni del Kernel.

1.3 Osservazioni

Osserviamo il risultato su un'immagine scelta casualmente del set creato e sulle due immagini aggiuntive.

Ricordiamo che più è alto il valore del PSNR maggiore sarà la vicinanza dell'immagine corrotta rispetto alla versione originale.

1.3.1 Analisi immagine geometrica

Analizziamo l'immagine img8.png al variare del valore σ con Dev.Std = 0.05:

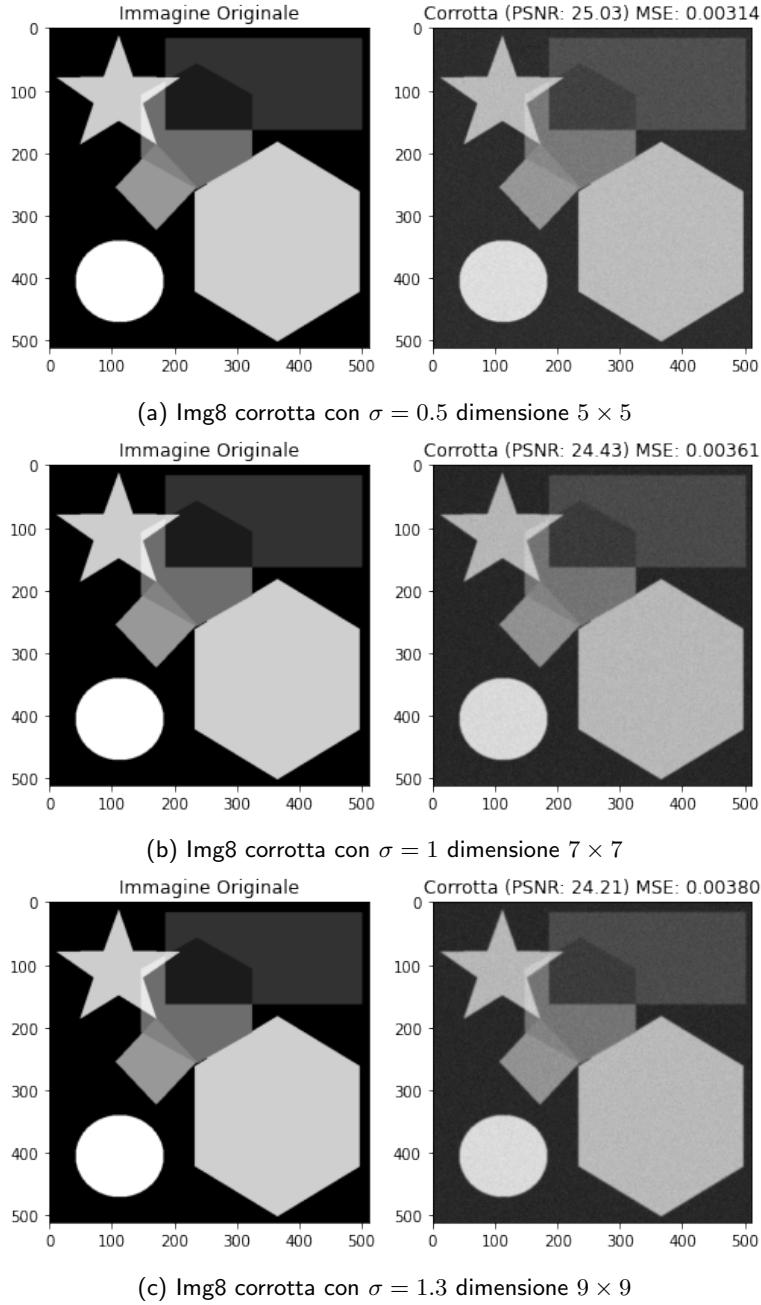
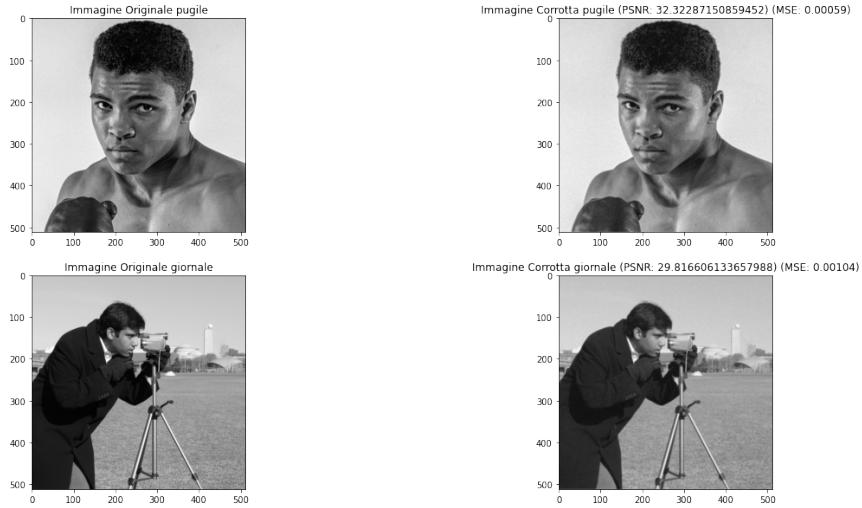


Figura 19: Immagine geometrica corrotta al variare di σ

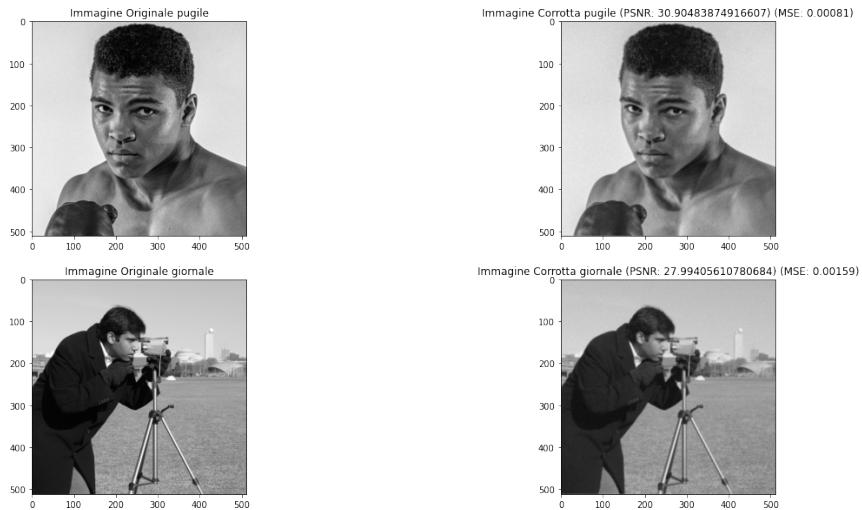
Le figure di sinistra rappresentano l'immagine originale, invece a destra sono riportate le immagini corrotte con i rispettivi valori di PSNR e MSE. Notiamo che all'aumentare delle dimensioni di sigma il valore di PSNR diminuisce che denota un peggioramento della qualità dell'immagine, infatti le immagini subiscono un'appiattimento dell'intensità della scala dei colori e i contorni delle varie figure geometriche perdono di fermezza.

1.3.2 Analisi immagini fotografiche

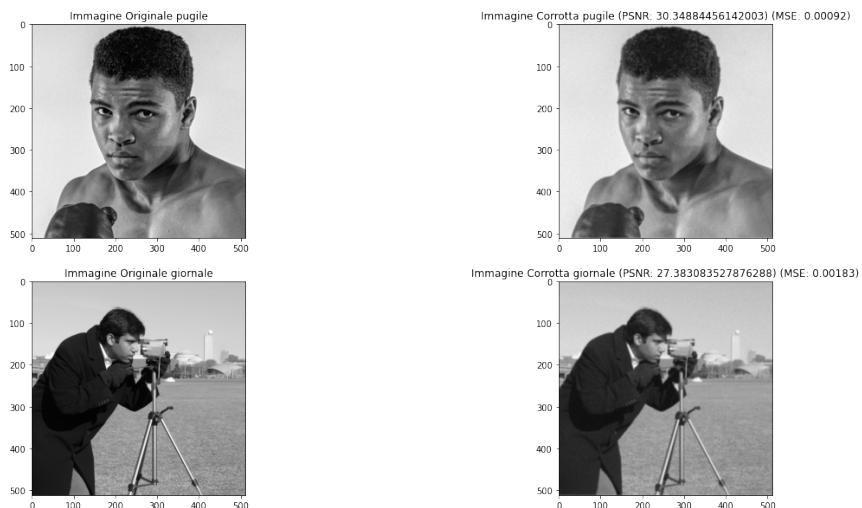
Analizziamo le immagini fotografiche al variare del valore σ con Dev.Std 0.05:



(a) immagini corrotte con $\sigma = 0.5$ dimensione 5×5



(b) immagini corrotte con $\sigma = 1$ dimensione 7×7



(c) immagini corrotte con $\sigma = 1.3$ dimensione 9×9

Figura 20: Immagine fotografica corrotta al variare di σ

Si nota un'altra volta che all'aumentare delle dimensioni di σ diminuisce il PSNR e l'immagine perde di incisività, le versioni corrotte benché risultino visivamente peggiori, si riesce ancora a ben distinguere il soggetto in primo piano, anche se sfocato, in tutte le immagini.

2 Ricostruzione di un immagine rispetto una versione corrotta

Una possibile ricostruzione dell'immagine originale x partendo dall'immagine corrotta b è la soluzione naïve data dal minimo del seguente problema di ottimizzazione:

$$x^* = \arg \min_x \frac{1}{2} \|Ax - b\|_2^2$$

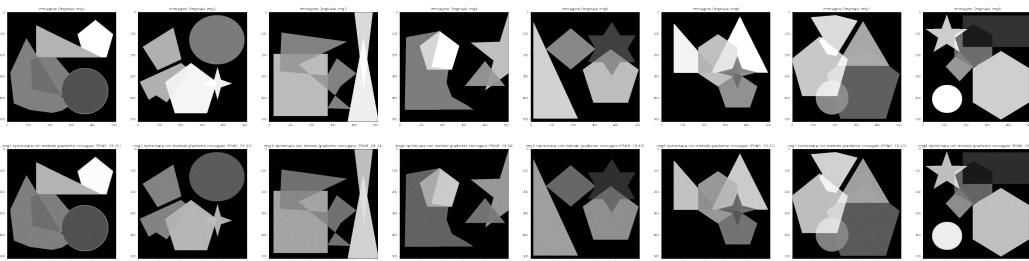
Nell'analisi delle immagini ricostruite useremo $\sigma = 1$ dimensione 7×7 come valore standard.

2.1 Metodo del Gradiente Coniugato (naive)

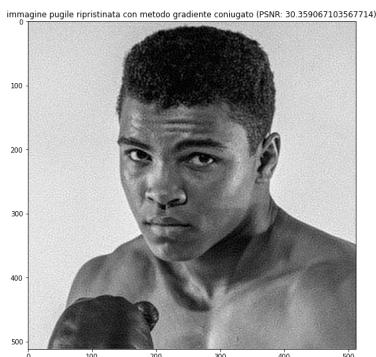
Il metodo del gradiente coniugato è un algoritmo per la risoluzione numerica di un sistema lineare la cui matrice sia simmetrica e definita positiva e consente di risolvere il sistema in un numero di iterazioni che è al massimo n .

La funzione f da minimizzare è data dalla formula $f(x) = \frac{1}{2} \|Ax - b\|_2^2$, il cui gradiente ∇f è dato da $\nabla f(x) = A^T Ax - A^T b$.

Utilizzando il metodo del gradiente coniugato implementato dalla funzione `minimize` abbiamo calcolato la soluzione naïve.



(a) Immagini geometriche ripristinate



(b) Immagine fotografica ripristinata



(c) Immagine con testo ripristinata

Figura 21: Immagini analizzate ripristinate con il Metodo del Gradiente Coniugato

```

1  from scipy.optimize import minimize
2  def f(x, B, labda = 0):
3      X = x.reshape(m,n)
4      res = 0.5*(np.linalg.norm(A(X, K)-B))**2 + 0.5*labda*(np.linalg.norm(X))**2
5      return np.sum(res)
6  def df(x, B, labda=0):
7      X = x.reshape(m,n)
8      res = AT(A(X, K)-B, K) + labda*X
9      RES = np.reshape(res, m*n)
10     return RES
11
12 func = lambda x: f(x, B) # o exec
13 grad_func = lambda x: df(x, B)
14 res = minimize(func, np.zeros(b.shape), method="CG", jac=grad_func, options={
15     'disp': True, 'maxiter':6})
16 RES = res.x.reshape(m,n)
17 PSNR = metrics.peak_signal_noise_ratio(X, RES) #alfabeto[4] solo lettere
18 plt.figure(figsize=(20,15))
19 plt.title(f'Ripristinata con metodo gradiente coniugato (PSNR: {PSNR})')
20 plt.imshow(RES, cmap="gray", vmin=0, vmax=1)

```

Figura 22: Codice in Python 3 del Metodo del Gradiente Coniugato applicato ad una singola immagine

Con il metodo del gradiente coniugato abbiamo ripristinato le immagini, ottenendo un PSNR definitivamente superiore rispetto al PSNR dell'immagine corrotta iniziale.

Per esempio considerando l'immagine fotografica avremo che:

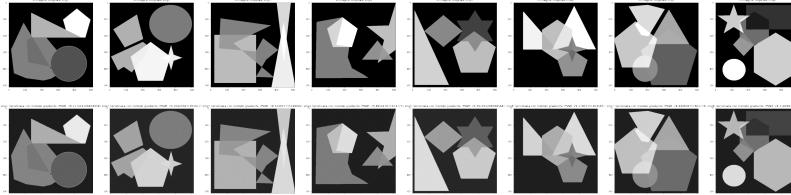
Immagine ripristinata 21b PSNR: 30.359067

Immagine corrotta 20b PSNR: 30.9049

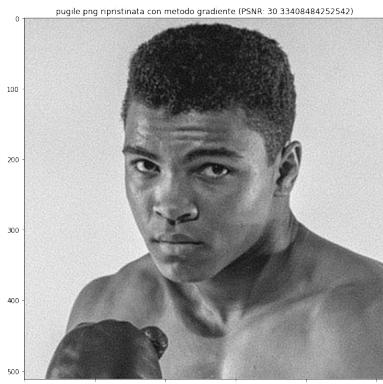
2.2 Metodo del Gradiente

Il metodo del gradiente è un algoritmo che calcola il vettore di minimo globale, ovvero: Un vettore x^* è un punto di minimo globale di $f(x)$ se $f(x^*) \leq f(x) \forall x \in R^n$.

Analogamente, un vettore x^* è un punto di minimo globale in senso stretto di $f(x)$ se $f(x^*) < f(x) \forall x \in R + nx \neq x^*$.



(a) Immagini geometriche ripristinate



(b) Immagine fotografica ripristinata



(c) Immagine con testo ripristinata

Figura 23: Immagini analizzate ripristinate con il Metodo del Gradiente

Algorithm 1 Metodo del Gradiente in pseudocode

```

1:  $x_0 \in R^n, k = 0$ 
2: while  $\nabla f(x_k) \neq 0$  and  $k < \text{maxit}$  do
3:   calcolare la direzione di discesa  $d_k$ 
4:   calcolare il passo di discesa  $\alpha_k$ 
5:    $x_{k+1} = x_k + \alpha_k d_k$ 
6:    $k = k + 1$ 
7: end while

```

▷ maxit è il massimo di iterazioni

```

● ● ●
1 listaValGradG = []
2 listaErrRelG = []
3 listaPSNRG = []
4 listaValFunG = []
5
6 def next_step(x, grad, B, labda = 0):
    # backtracking procedure for the choice of the steplength
7     alpha=1.1
8     c1 = 0.25
9     p=-grad
10    j=0
11    jmax=10
12
13    #condizioni che servono per soddisfare dei criteri di convergenza - condizioni di Wolfe
14    while ((f(x + alpha*p, B, labda) > f(x, B, labda)+c1*alpha*grad.T@p) and j <jmax):
15        alpha = alpha*0.5
16        j+=1
17    return alpha
18
19 def gradient_minimize(B, labda= 0, maxit=19, abstopp = 1.e-6):
20     x_last =np.zeros(m*n)
21     k = 0
22     while (np.linalg.norm(df(x_last, B, labda))>abstop and k < maxit):
23         k=k+1
24         grad = df(x_last, B, labda)
25         step = next_step(x_last, grad, B, labda)
26
27         listaValFunG.append(f(x_last, B))
28         x_last=x_last-step*grad
29
30         listaValGradG.append(np.linalg.norm(grad.reshape(m,n), "fro"))
31         listaErrRelG.append(np.linalg.norm(x_last.reshape(m,n) - X, "fro")/np.linalg.norm(X, "fro"))
32         listaPSNRG.append(metrics.peak_signal_noise_ratio(X, x_last.reshape(m,n)))
33
34     z_naive = gradient_minimize(B, labda = 0, maxit = 50)
    #maxit = 27. Messo maxit = 50 per studiare metodi e loro velocità
35     PSNR = metrics.peak_signal_noise_ratio(X, z_naive.reshape(m,n))
    #cambiare anche qui ogni volta nome immagine originale
36     plt.figure(figsize=(30,10))
37     plt.title(f'Ripristinata con metodo gradiente (PSNR: {PSNR})')
38     plt.imshow(z_naive.reshape(m,n), cmap="gray")
39

```

Figura 24: Codice in `Python 3` del metodo del gradiente applicato ad una singola immagine

Notiamo anche qui che l'immagine ripristinata ha un PSNR molto più alto rispetto all'immagine corrotta. Questo risulta in un'immagine con una qualità corretta e simile all'immagine originale.

Per esempio considerando l'immagine fotografica avremo che:

Immagine ripristinata 23b PSNR: 30.334085

Immagine corrotta 20b PSNR: 30.9049

2.3 Metodo del Gradiente e Metodo del Gradiente Coniugato a confronto

Notiamo che tra i due metodi che il primo ci dà come risultato delle immagini con un PSNR definitivamente più alto rispetto al secondo, le immagini sono qualitativamente più simili alle immagini originali.

Analizziamo il comportamento del gradiente utilizzando i due metodi sull'immagine `data.camera()`:

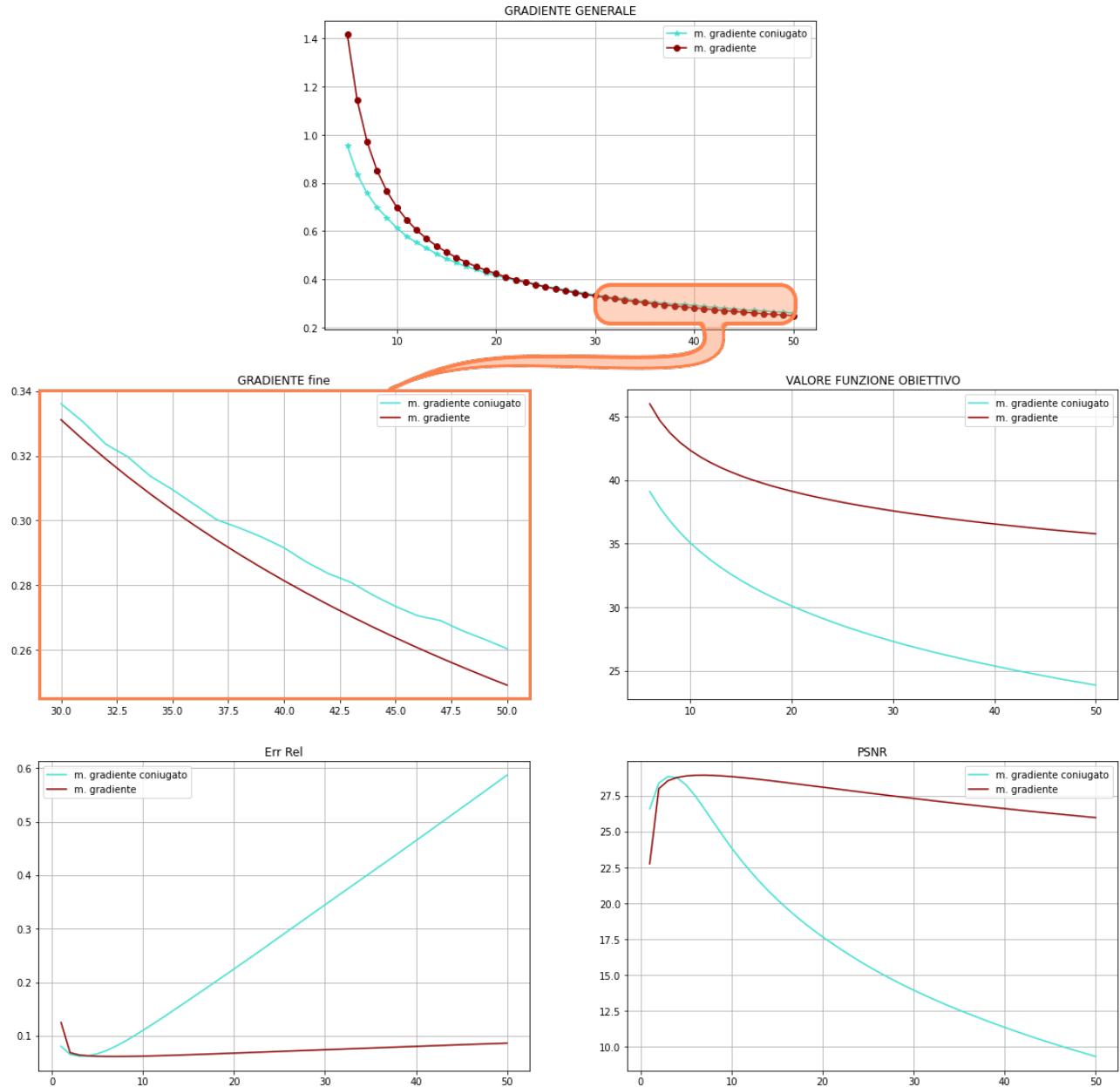


Figura 25: Gradiente immagine `data.camera()`

Per quanto riguarda l'immagine fotografica `pugile.png`, abbiamo riscontrato:

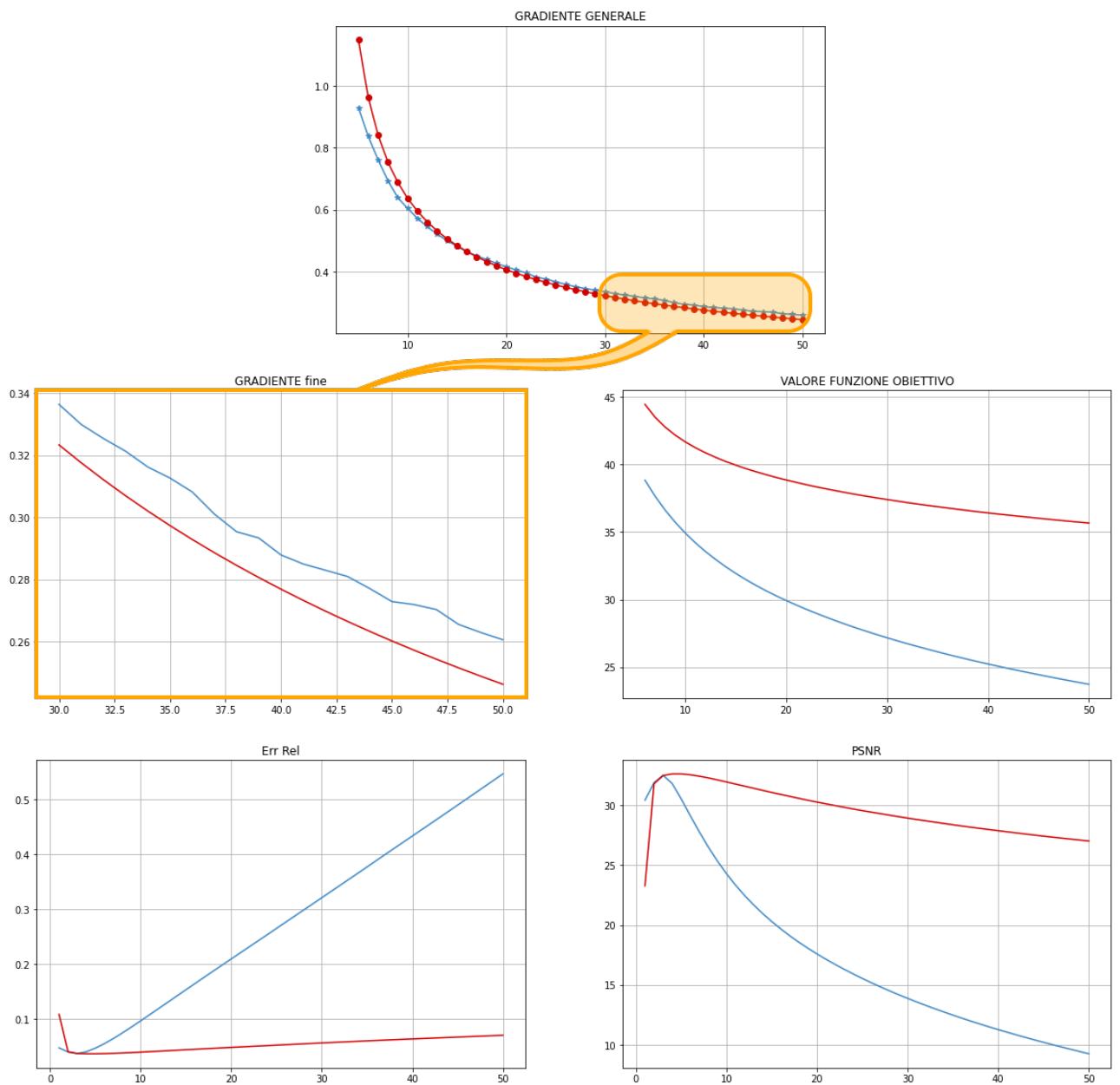


Figura 26: Gradiente immagine fotografica

3 Metodi di Regolarizzazione

I metodi di regolarizzazione rinunciano a trovare la soluzione esatta del problema del precedente problema di ottimizzazione, ma invece calcolano la soluzione di un problema leggermente diverso ma meglio condizionato. Quest'ultimo viene chiamato problema regolarizzato.

3.1 Metodo di Regolarizzazione con Metodo del Gradiente minimize

Vediamo che al crescere del λ , il PSNR fino a $\$lambda \simeq 0.02$ continua ad aumentare, quindi migliorando l'immagine, raggiunge un picco massimo e successivamente decresce.

3.1.1 Immagini geometriche regolarizzate

Analizziamo i grafici ottenuti cercando di ridurre il rumore nella ricostruzione delle immagini del dataset.

Grafici andamento PSNR, media PSNR e MSE nella ricostruzione delle immagini con il metodo di regolarizzazione

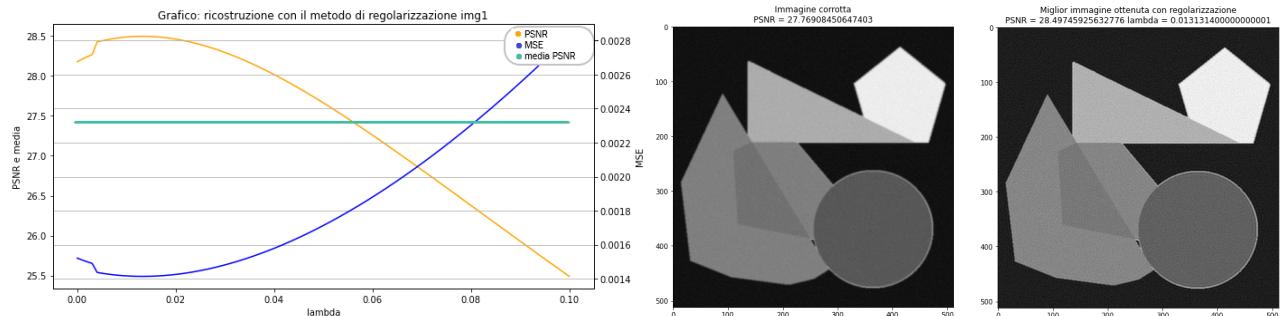


Figura 27: Ricostruzione immagine geometrica img1.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

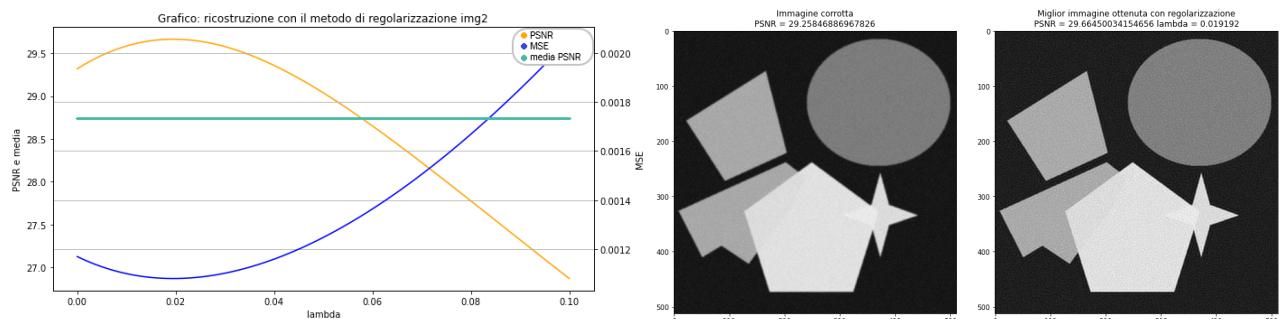
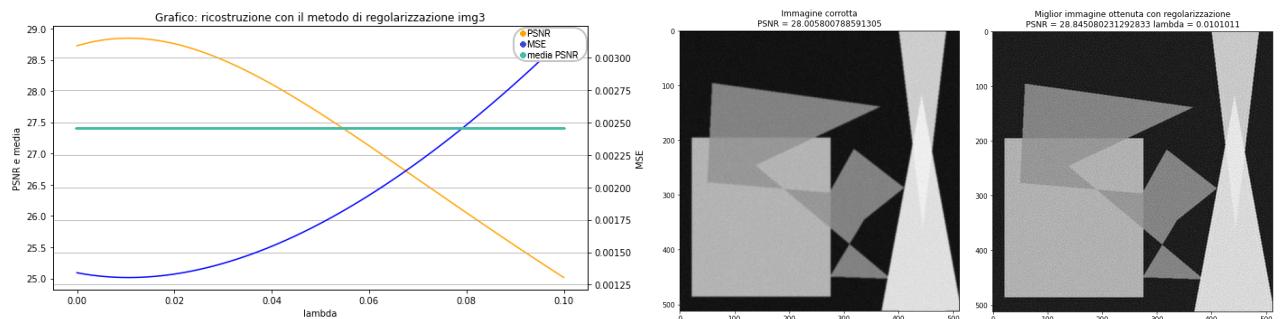


Figura 28: Ricostruzione immagine geometrica img2.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]



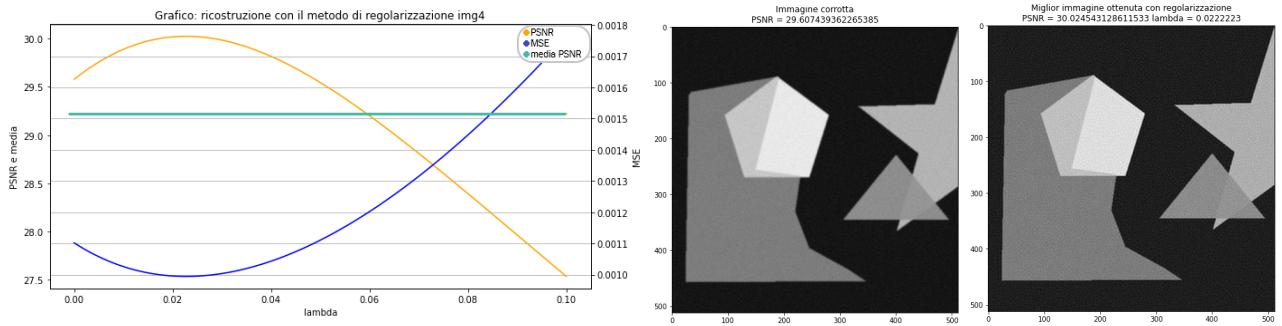


Figura 30: Ricostruzione immagine geometrica img4.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

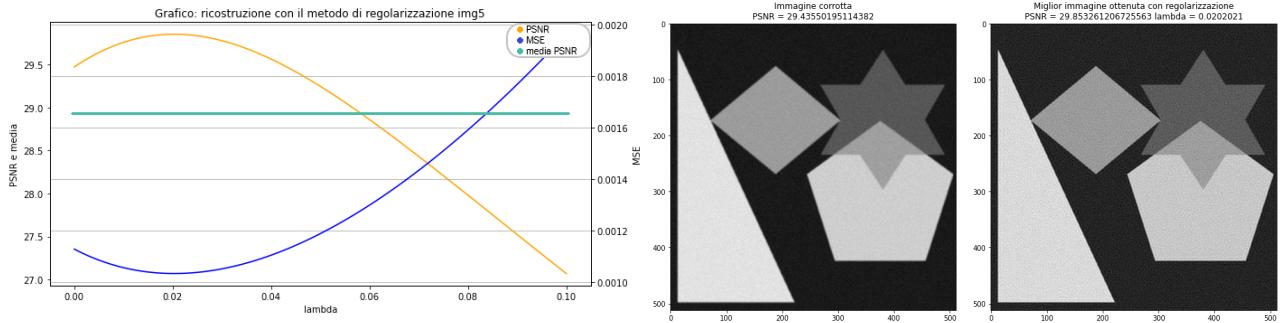


Figura 31: Ricostruzione immagine geometrica img5.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

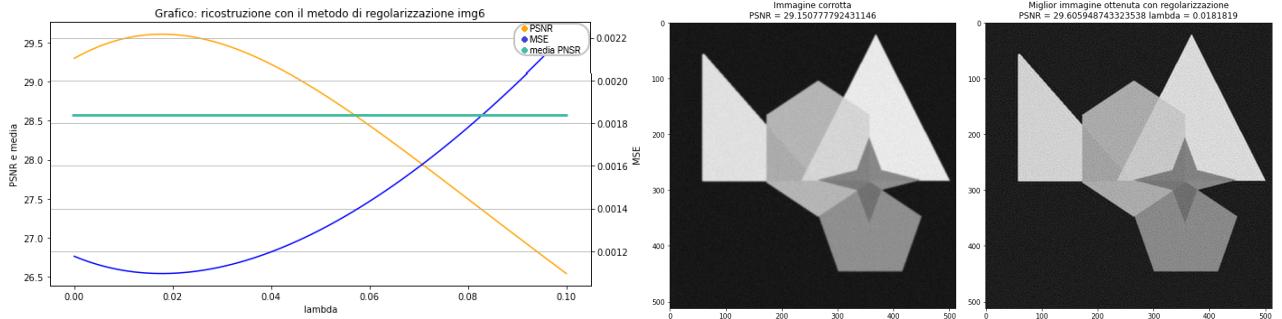


Figura 32: Ricostruzione immagine geometrica img6.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

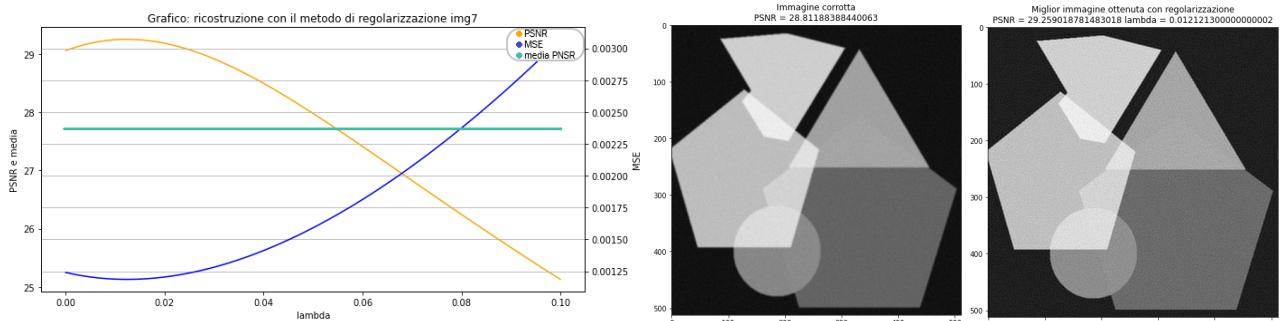


Figura 33: Ricostruzione immagine geometrica img7.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

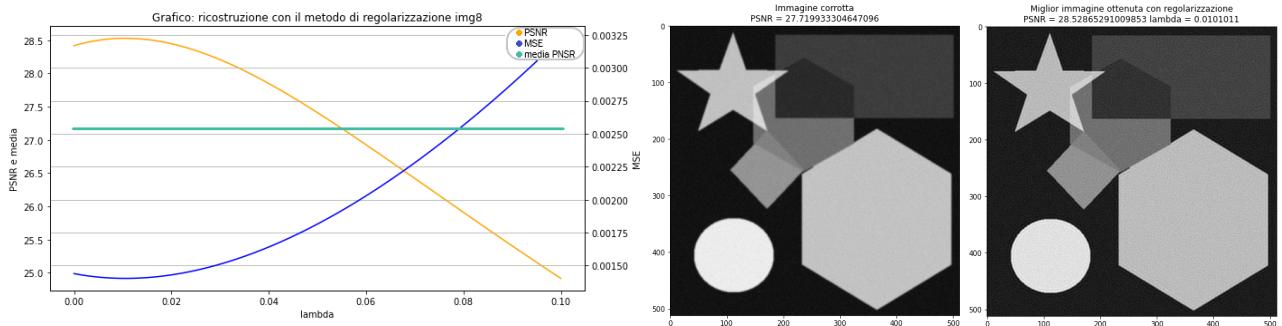


Figura 34: Ricostruzione immagine geometrica img8.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

3.1.2 Immagini fotografiche regolarizzate

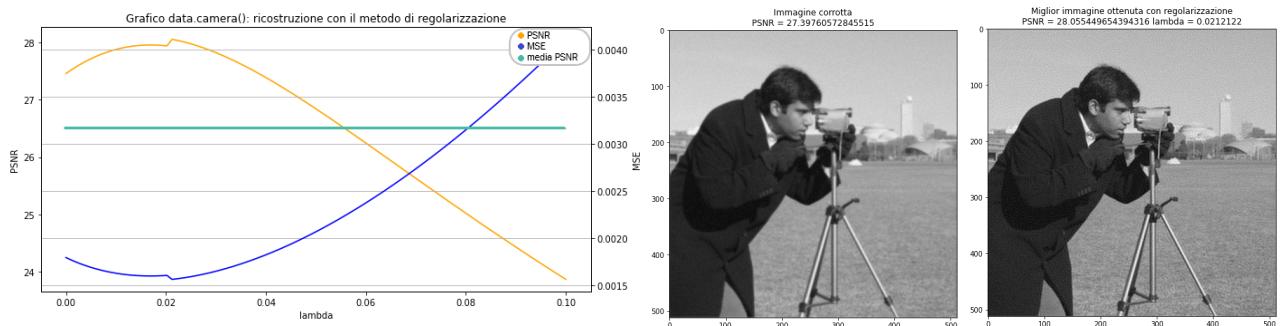


Figura 35: Ricostruzione immagine fotografica data.camera() [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

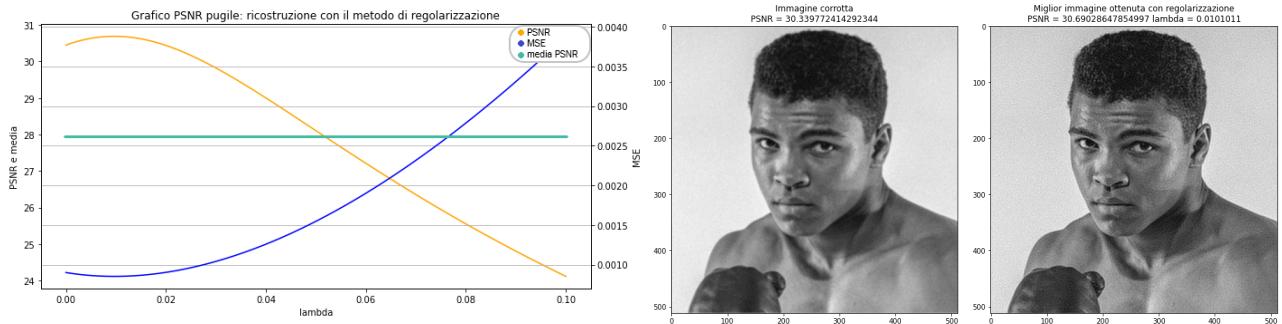


Figura 36: Ricostruzione immagine fotografica pugile.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

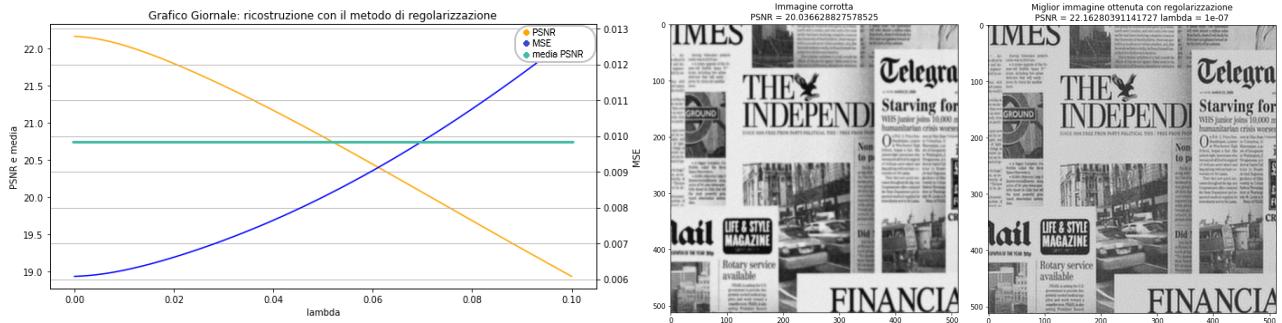


Figura 37: Ricostruzione immagine fotografica giornale.png [a destra: immagine corrotta, sinistra: miglior immagine ottenuta con regolarizzazione]

3.2 Metodo di Regolarizzazione di Tikhonov

Per ridurre gli effetti del rumore nella ricostruzione è necessario introdurre un termine di regolarizzazione di Tikhonov.

Si considera quindi il seguente problema di ottimizzazione:

Si deve risolvere $Ax_\epsilon = b_\epsilon$ con $b_\epsilon = b + \epsilon$. Al posto di risolvere direttamente il sistema lineare (se il sistema è quadrato) o di minimizzare la norma 2 del residuo (se il sistema è rettangolare) $\|Ax_\epsilon - b_\epsilon\|_2^2$,

si aggiunge un vincolo di regolarità alla soluzione e si minimizza, ad esempio

$$\|Ax_\epsilon - b_\epsilon\|_2^2 + \gamma_\epsilon \|x_\epsilon\|_2^2$$

che rappresenta la **forma standard** della regolarizzazione di Tikhonov.

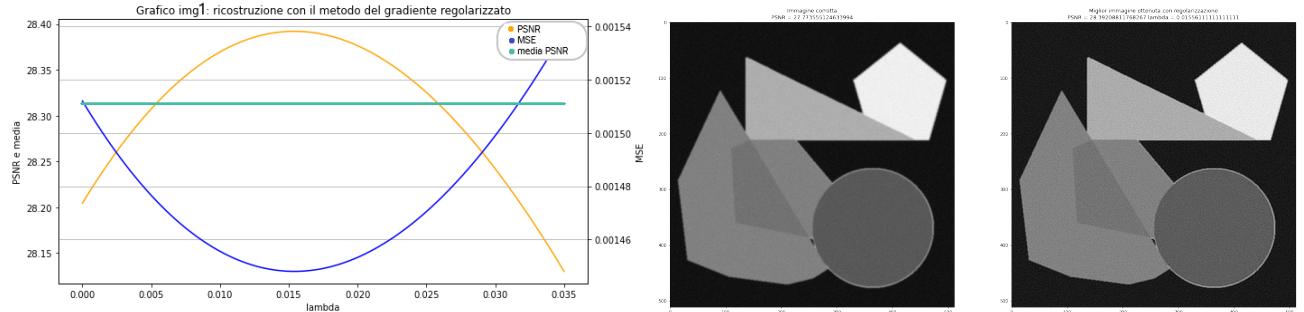


Figura 38: Ricostruzione immagine geometrica img1.png

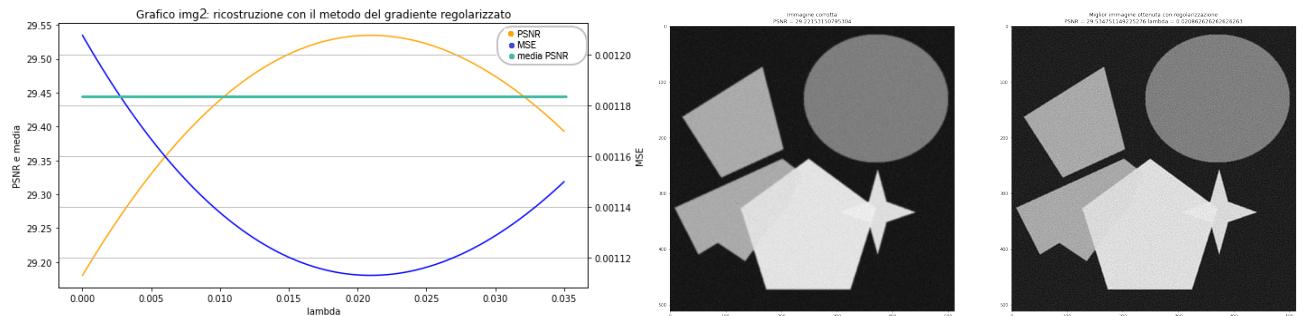


Figura 39: Ricostruzione immagine geometrica img2.png

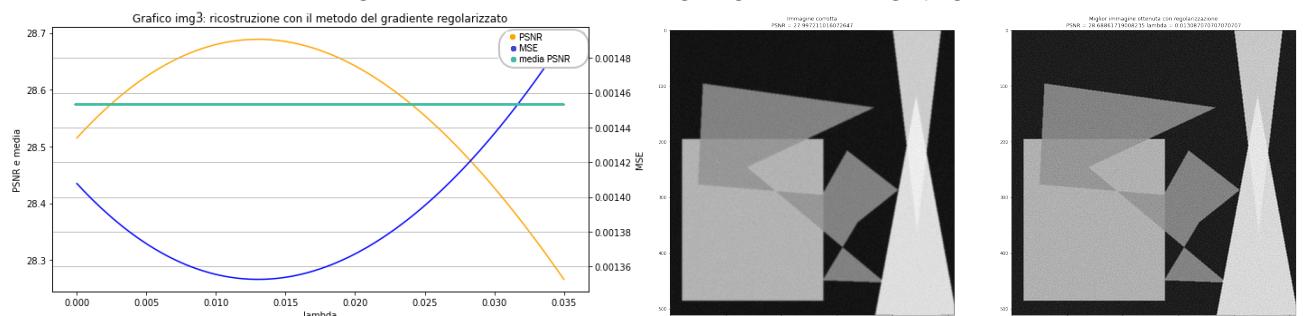


Figura 40: Ricostruzione immagine geometrica img3.png

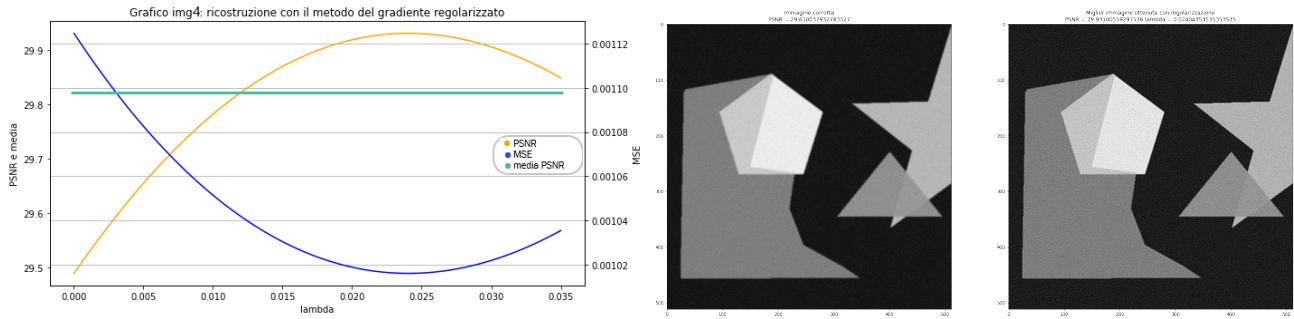


Figura 41: Ricostruzione immagine geometrica img4.png

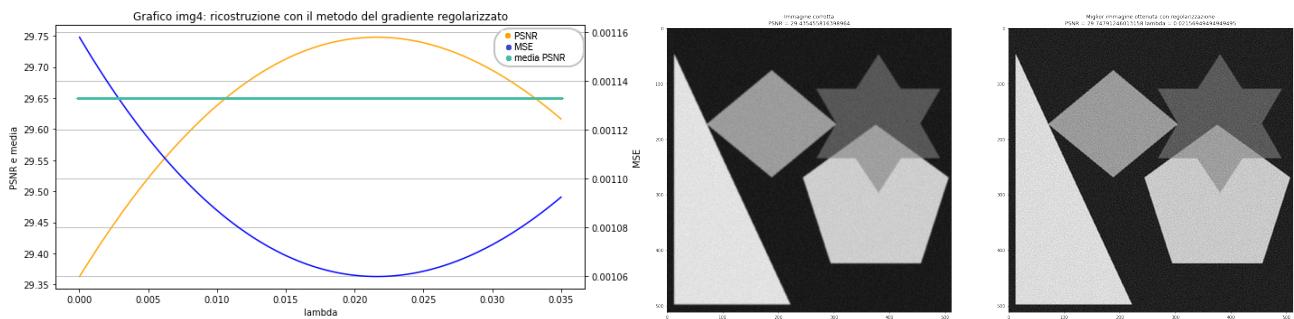


Figura 42: Ricostruzione immagine geometrica img5.png

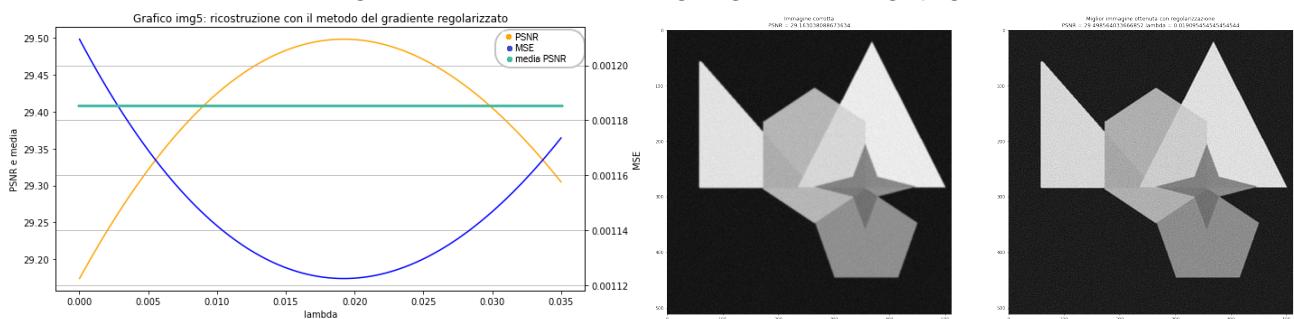


Figura 43: Ricostruzione immagine geometrica img6.png

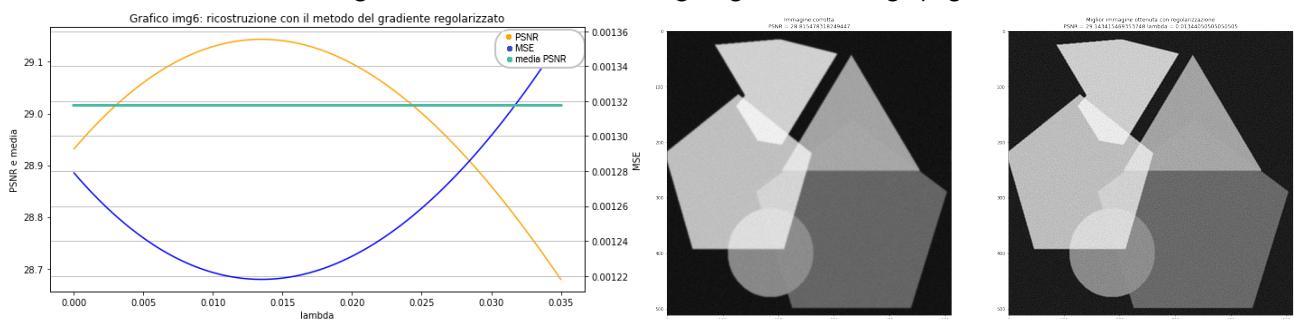


Figura 44: Ricostruzione immagine geometrica img7.png

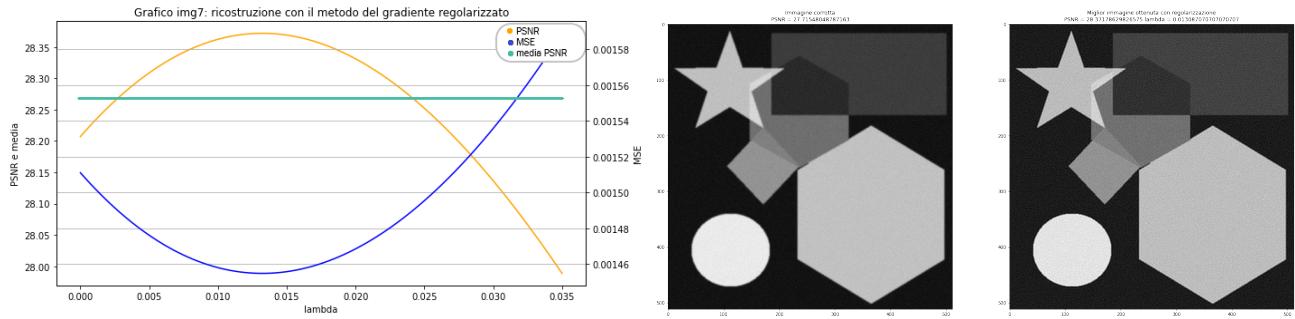


Figura 45: Ricostruzione immagine geometrica img8.png

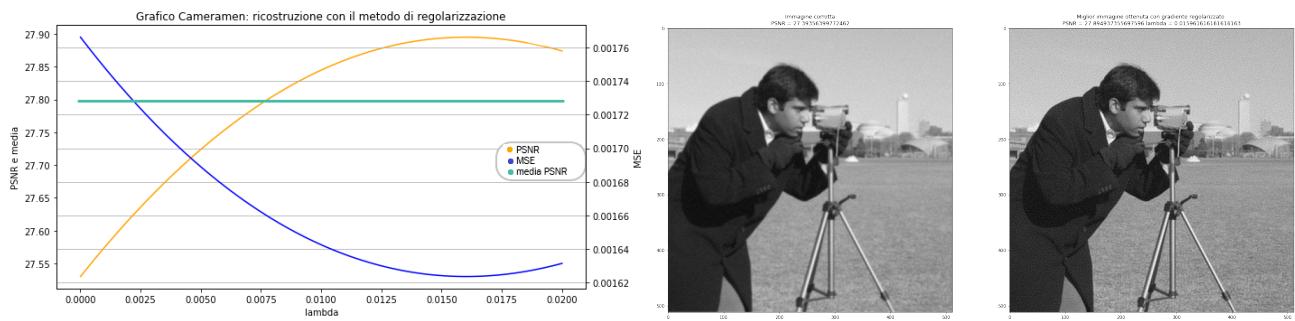


Figura 46: Ricostruzione immagine fotografica data.camera()

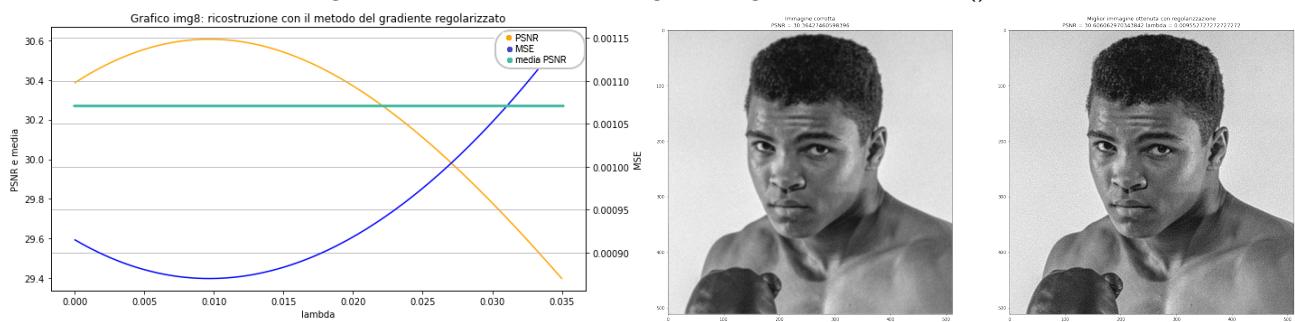


Figura 47: Ricostruzione immagine fotografica pugile.png

4 Variazione Totale

Tramite un algoritmo possiamo recuperare immagini sfocate basandoci sulla Variazione totale partendo da una Blurring Point-Spread function di un'immagine.

La variazione totale è definita dalla seguente formula:

$$TV(u) = \sum_i^n \sum_j^m \sqrt{\|\nabla u(i, j)\|_2^2 + \epsilon^2}$$

Per calcolare il gradiente dell'immagine ∇u usiamo la funzione `np.gradient` che approssima la derivata per ogni pixel calcolando la differenza tra pixel adiacenti.

I risultati sono due immagini della stessa dimensione dell'immagine in input, una che rappresenta il valore della derivata orizzontale dx e l'altra della derivata verticale dy . Il gradiente dell'immagine nel punto (i, j) è quindi un vettore di due componenti, uno orizzontale contenuto in dx e uno verticale in dy .

$$x^* = \arg \min_x \frac{1}{2} \|Ax - b\|_2^2 + \lambda TV(u)$$

il cui gradiente ∇f è dato da:

$$\nabla f(x) = (A^T Ax - A^T b) + \lambda \nabla TV(x)$$

4.1 Variazione totale dell'immagine fotografica pugile:

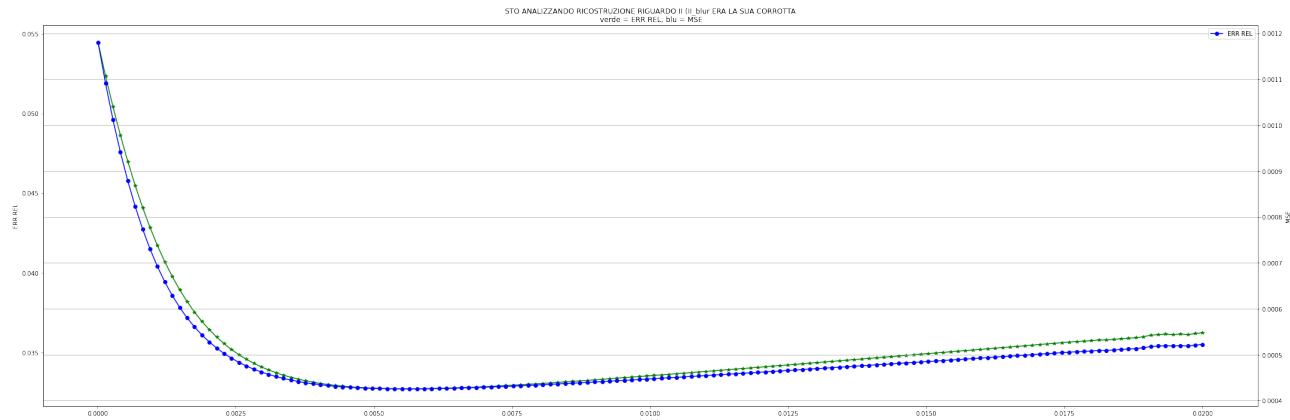


Figura 48: in verde: Errore Relativo, in blu: MSE

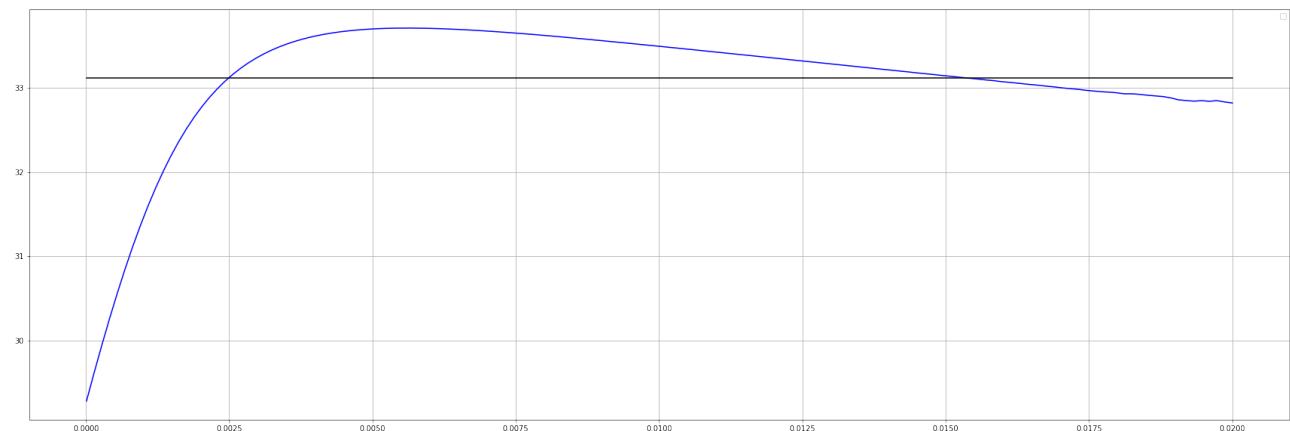


Figura 49: PSNR

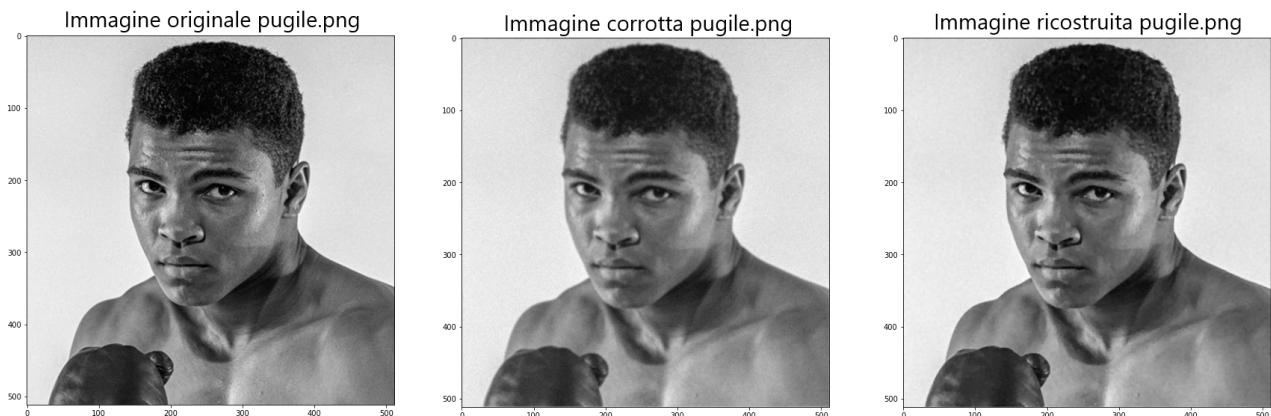


Figura 50: Immagine Originale, Immagine Corrotta, Immagine Ricostruita

4.2 Variazione totale dell'immagine con testo:

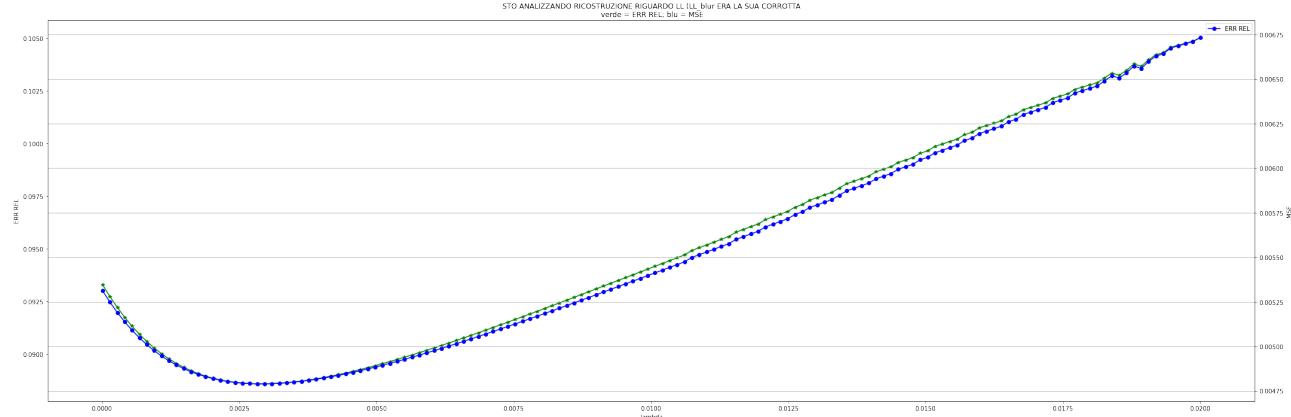


Figura 51: in verde: Errore Relativo, in blu: MSE

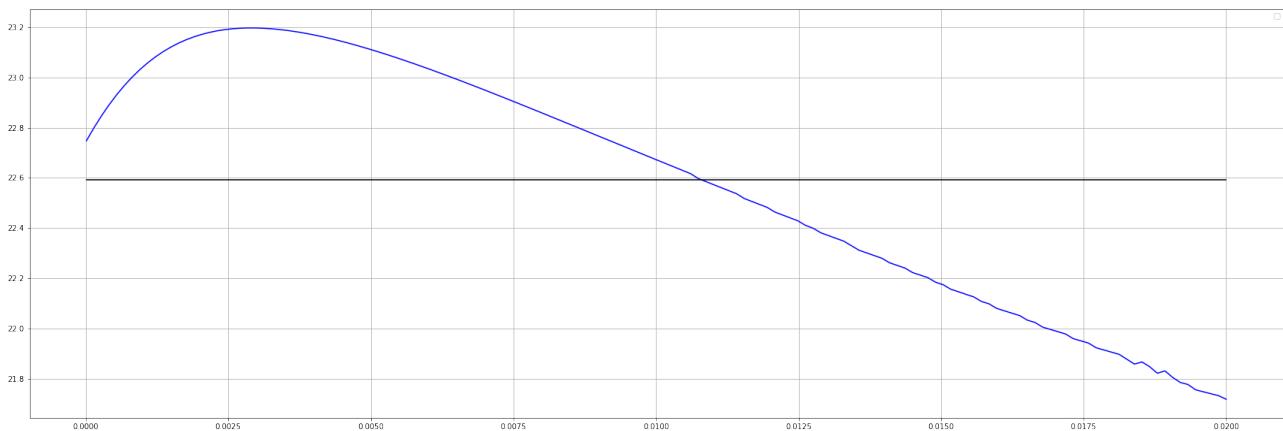


Figura 52: PSNR



Figura 53: Immagine Originale, Immagine Corrotta, Immagine Ricostruita

5 Conclusioni