Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

---

Tesi di Laurea Triennale in Ingegneria Informatica

Metodi di deconvoluzione di immagini affette da motion blur

Candidato
Guglielmo Menchetti

Relatore
Prof. Fabrizio Argenti

Anno Accademico 2015-2016

Indice

Introduzione

Lo sviluppo dell'*image restoration* ha una lunga storia che ha inizio a seguito dei primi programmi spaziali organizzati dagli Stati Uniti e dall'Unione Sovietica negli anni '50-'60. Durante le varie missioni venivano scattate fotografie della terra e del sistema solare che, per quegli anni, avevano un valore inestimabile.

Tuttavia le immagini catturate risultavano degradate. Le cause erano molteplici, per esempio il rapido cambiamento di rifrazione dell'amosfera, il tempo di esposizione lento in relazione al movimento della navicella spaziale oppure il rumore atmosferico.

Per inciso, considerazioni simili riguardo alla degradazione delle immagini possono essere fatte anche in altri campi scientifici, come quello della medicina, per quanto riguarda le immagini al microscopio o ai raggi-X.

Negli ultimi anni, grazie allo sviluppo della tecnologia digitale, le immagini digitali hanno raggiuto un livello di qualità veramente notevole. Ciononostante anche oggi capita che le immagini acquisite non riproducano fedelmente lo scenario reale, ma continuino a presentare una degradazione della qualità, o *blur*.

Generalmente, i tipi di *blur* più comuni due: il *defocus* (o *out of focus*) e il *motion blur*. Con il primo si intende che l'immagine, o una sua parte, presenta una scarsa nitidezza, mentre con il secondo che il soggetto della foto

risulta allungato e sfuocato.

Con il termine *image restoration* si indica quindi l'operazione di prendere un'immagine deteriorata ed intervenire per il blur e il *rumore*.

Con questo progetto di tesi andremo, inizialmente ad approfondire il concetto di *motion blur lineare*: ne verrà data una definizione matematica e verrà poi implementato un metodo per rimuovere tale degradazione dalle immagini. In particolare, è stato pensato di approfondire il concetto di *rappresentazione sparsa* dei segnali, un argomento molto studiato a partire dai primi anni Duemila, che si concretizza con l'utilizzo della *trasformata Wavelet*.

Verrà quindi sviluppato un metodo per la ricostruzione dell'immagine con due approcci: il primo, detto *non-blind* è applicabile quando siamo a conoscenza di alcuni parametri del *blur*, mentre il secondo, chiamato *blind*,viene usato quando del blur non è disponibile alcuna informazione.

Infine, verranno effettuati dei test su immagini la cui degradazione è stata inserita artificialmente, applicando entrambe i metodi di restaurazione.

La struttura della tesi è la seguente:

Nel **capitolo 1** verrà definito il modello matematico del problema e si descriveranno alcuni metodi classici di *deblurring*.

Nel **capitolo 2** verrà approfondito il concetto di *sparsità* per le immagini, con riferimento alla trasformata Wavelet, e verrà descritto il problema di deconvoluzione nel caso di rappresentazione sparsa.

Nel **capitolo 3** verranno descritte le tecniche utilizzate per la risoluzione del *deblurring*, *non-blind* e *blind*.

Nel **capitolo 4** si riporteranno i risultati ottenuti con l'esecuzione dei test sperimentali e nel **capitolo conclusivo**, verranno riportate alcune considerazioni discutendo brevemente dei possibili sviluppi del progetto.

Infine è stata inserita un'**Appendice** dove sarà possibile trovare alcuni ap-

profondimenti sulle trasformate utilizzate.

Capitolo 1

Il Problema del *Blur*

Un'immagine affetta da *blur* risulta poco definita e poco dettagliata. L'effetto risultante è quello di non poter percepire in modo corretto tutti gli elementi e le loro forme. La forma di un oggetto è infatti definita dai bordi di quest'ultimo. In presenza di *blur*, è presente una riduzione del contenuto dei bordi che rende la transizione da un colore ad un altro *sfumato* (*smooth*).

Il principale obiettivo delle tecniche di restauro è ovviamente quello di rendere migliore un'immagine cercando di ripristinarne il contenuto informativo. Il restauro tenta quindi di riparare un'immagine degradata facendo uso di una conoscenza *a priori* del fenomeno che ha provocato il degrado. Le tecniche di restauro sono quindi orientate alla modellizzazione del processo di degrado nel tentativo di riuscire a generare il processo inverso, in grado di ricostruire l'immagine originale.

In questo paragrafo verrà presentata una definizione del modello matematico del *blur* e una discussione dei metodi detti "stato dell'arte" (*state of the art methods*) di restauro dell'immagine.

1.1 Modello Matematico

Il problema del *blur* è modellato come un processo di convoluzione tempo-invariante tale che

$$f(x, y) = g(x, y) * p(x, y) + n(x, y) \tag{1.1}$$

dove

- $*$ rappresenta l'operatore discreto di *convoluzione*
- *g(x,y)* è l'*immagine latente* da recuperare
- *p(x,y)* rappresenta il *blur kernel* (*funzione di degradazione* o *point spread function (PSF)*)
- *n(x,y)* rappresenta il *rumore*

Il problema di recuperare l'immagine latente g da quella osservata f viene definito *deconvoluzione* (*image deconvolution*).

L'Immagine

Un'immagine può essere definita come una funzione bidimensionale,

$$f(x, y)$$

dove x e y sono le *coordinate spaziali* (sul piano) e l'ampiezza di f, in ogni coppia di coordinate (x, y), viene chiamata *intensità* o livello di grigio dell'immagine in quel determinato punto.

Quando x, y e i valori dell'ampiezza di f sono quantità discrete, possiamo definire l'immagine come un'**immagine digitale**: questa è composta da un numero finito di elementi, ciascuno dei quali ha una particolare posizione e un determinato valore. Questi elementi sono detti *pixel*. Il passaggio da una rappresentazione in continua dell'immagine a una discreta si ha tramite l'operazione di *campionamento*.

Generalmente un'immagine

$$f(x, y) \in \mathcal{R}^{M \times N}$$

, viene rappresentata tramite una matrice di M righe e N colonne.[1]

Il Rumore

Le principali sorgenti di rumore nelle immagini digitali si presentano durante il processo di acquisizione e/o trasmissione. La resa dei *sensori di imaging* è influenzata da una varietà di fattori, come le condizioni ambientali durante l'acquisizione e la qualità degli stessi elementi di cattura. Le immagini si possono anche deteriorare durante la trasmissione, principalmente a causa di interferenze nel canale usato per la trasmissione stessa.

Di rilevante importanza sono i parametri spaziali del rumore e l'eventuale correlazione di questo con l'immagine. Le proprietà in frequenza del rumore si riferiscono al contenuto nel *dominio di Fourier*. In particolare, quando lo *spettro di Fourier* del rumore è costante, questo è definito *rumore bianco*.

La componente di rumore viene considerata una variabile casuale, caratterizzata dalla propria *funzione di densità di probabilità* (o *PDF*).

Nella pratica, per la loro duttilità matematica - sia nel dominio spaziale che in quello della frequenza - vengono spesso utilizzati i modelli di rumore *gaussiani* (o *normali*).

La PDF di una variabile casuale gaussiana z è data da

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(z-m)^2}{2\sigma^2}}$$

dove z rappresenta l'*intensità*, m è il *valore medio* di z e σ è la sua *deviazione standard*.[1][1]

[1]Il valore σ^2 è detta *varianza* di z.

La Funzione di Degradazione (PSF) e il *Motion Blur*

I tipi di *blur* da cui può essere affetta l'immagine sono molteplici e variano in base alla funzione di degradazione p applicata, tramite una convoluzione, all'immagine originale p.

Uno dei tipi di *blur* più comuni è il *defocus* (o *out of focus*), ovvero lo sfuocamento dell'immagine, la tipica degradazione conosciuta da tutti coloro che utilizzano fotocamere, videocamere, microscopi o telescopi.

Più precisamente, il *defocus* riduce la nitidezza e il contrasto dell'immagine, cosicchè i dettagli della scena sono confusi o addirittura non percepibili.

Un altro tipo di *blur* molto diffuso, quello che vogliamo approfondire in questa tesi, è il *motion blur*.

Quando un'immagine è catturata da una fotocamera, questa non rappresenta solamente la scena in un singolo istante di tempo, ma quella in un periodo. Se gli elementi all'interno della scena si muovono velocemente, oppure durante il *tempo di esposizione*, gli oggetti dell'intera scena possono apparire degradati (*blurry*) a causa del loro movimento.

D'altra parte, causa di *motion blur*, è anche il movimento della fotocamera (*camera shake*): in particolare, la degradazione delle immagini avveniene quando vengono scattate fotografie con un lungo tempo di esposizione ed una scarsa condizione di luce. In tal caso si parla di *uniform motion blur* perché tutti gli elementi della scena risultano uniformemente degradati.

In questa tesi viene utilizzata un'approssimazione lineare di tale effetto (*linear uniform motion blur*). La PSF in questo caso rappresenta una media locale e uniforme dei pixel vicini, cioè può essere modellata nel seguente modo:

$$p(x,y) = \begin{cases} \frac{1}{L} & \text{se} \quad \sqrt{x^2+y^2} \leq \frac{L}{2} \\ 0 & \text{altrimenti} \end{cases} \tag{1.2}$$

dove

$$y/x = \tan\theta$$

Il *motion blur* è quindi regolato da due parametri, la *lunghezza* del blur L e l'*angolo* θ.[**BlurEstimationLength**]

Nel capitolo (3) verrà proposto un metodo per stimare tali parametri in modo tale da poter ricostruire la PSF che ha generato il *blur*.

1.2 Deconvoluzione *Blind* e *Non-Blind*

Il problema della deconvoluzione si può distinguere in due categorie, in base alla disponibilità della PSF (p).

Se la PSF è disponibile prima della deconvoluzione questa viene detta *non-blind*. Si tratta di un problema di inversione *mal-condizionato* (*ill-conditioned*) in quanto la ricostruzione dell'immagine è molto suscettibile di errore dal momento che una piccola variazione del rumore (n) può comportare una pessima ricostruzione dell'immagine originale (g).

Se invece il *kernel* (p) non è disponibile, il problema della determinazione di g è detto *blind*: in questo caso, oltre all'immagine originale, deve essere stimata anche la PSF. Generalmente, una deconvoluzione *blind* è un problema di inversione *mal-condizionato* e *mal-posto* (*ill-posed*) perché, oltre alla sensibilità al rumore, il problema è *sotto-vincolato* (*underconstrained*) e può ammettere infinite soluzioni.

La rimozione del *motion blur* dalle immagini è generalmente di tipo *blind*. Per ridurre la complessità della *deconvoluzione blind*, in particolare del suo mal-condizionamento, si assume di conoscere alcune informazioni a priori sia sul *kernel* che sull'immagine da recuperare. In questa tesi è stata utilizzata

come conoscenza a priori la *sparsità* dell'immagine che verrà approfondita nel capitolo 3.

1.3 State Of the Art Methods

Alcune tecniche di restauro vengono meglio formulate nel dominio spaziale mentre altre si adattano meglio al dominio della frequenza. Generalmente, problematiche legate al *blurring* sono difficili da affrontare nel dominio spaziale. È quindi meglio scegliere opportuni filtri nel dominio della frequenza.

In questo paragrafo si descrivono i metodi "classici" per effettuare la deconvoluzione *non-blind*, cioè quando la funzione di degrado p è nota.

Soluzione Naïve

Una prima soluzione *naïve* si ha considerando la rappresentazione matriciale dell'equazione (1.1) tale che:

$$f = Hg + n \tag{1.3}$$

dove f, g ed n sono, rispettivamente, l'*immagine ossevata*, l'*immagine originale* e il *rumore additivo*. Queste componenti vengono rappresentate in *ordine lessicografico* (*lexicographical order*) ottenuto impilando le colonne della matrice dell'immagine, ottenendo un vettore.[2] La matrice $H \in \mathcal{R}^{MN \times MN}$ è una *matrice di Toeplitz a blocchi* e rappresenta l'operatore lineare di convoluzione.[3]

[2]Supponendo che l'immagine originale sia $g \in \mathcal{R}^{M \times N}$, si ottiene il vettore $g \in \mathcal{R}^{MN \times 1}$.

[3]La creazione di tale matrice verrà approfondita nel capitolo (3)

Supponendo quindi di conoscere tale matrice, un primo approccio *Naïve* è quello di ricostruire l'immagine latente, invertendo la matrice H, cioè:

$$g = H^{-1}f \tag{1.4}$$

Questo primo approccio fallisce, in particolare perché non viene considerato l'apporto dovuto al rumore. Considerando anche quest'ultimo, si ottiene:

$$g = H^{-1}f + H^{-1}n \tag{1.5}$$

dove il termine $H^{-1}n$ è detto *inverted noise* e misura il contributo negativo del rumore additivo nella ricostruzione dell'immagine: più questo è grande più la ricostruzione risulterà insoddisfacente.[2]

Filtraggio Inverso

Questo metodo rappresenta un approccio diretto per la risoluzione di (1.1) nel dominio della frequenza. Si consideri la *trasformata discreta di Fourier 2-D* dell'equazione (1.1):

$$F(u,v) = G(u,v)P(u,v) + N(u,v) \tag{1.6}$$

ottenuta come:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{1.7}$$

con

$$f(x,y) \in \mathcal{R}^{M\times N}$$

dove i valori

$$u = 0,\ldots,M-1$$

$$v = 0,\ldots,N-1$$

Con questo metodo viene calcolata la stima $\hat{G}(u,v)$ della trasformata dell'immagine originale, dividendo la trasformata dell'immagine degradata $F(u,v)$ per la funzione di degrado $H(u,v)$:

$$\hat{G}(u,v) = \frac{F(u,v)}{H(u,v)} \tag{1.8}$$

Sostituendo al membro sinistro dell'equazione (1.6) il valore $F(u,v)$ ricavato dall'equazione (1.8) si ottiene:

$$\hat{G}(u,v) = G(u,v) + \frac{N(u,v)}{H(u,v)} \tag{1.9}$$

Quest'ultima espressione afferma che, pur conoscendo la funzione di degrado, non possiamo ripristinare l'immagine non degradata (la trasformata inversa di Fourier di $G(u,v)$) perché $N(u,v)$ non è noto. Inoltre, se la funzione di degrado ha valori nulli o molto piccoli, il rapporto $\frac{N(u,v)}{H(u,v)}$ potrebbe facilmente essere dominante rispetto al valore di $\hat{G}(u,v)$.

Un metodo per ovviare al problema è quello di limitare le frequenze del filtro a valori vicini all'origine. Infatti si può far vedere che $H(0,0)$ è di solito il valore più alto di $H(u,v)$ nel dominio della frequenza. Quindi, limitando l'analisi a frequenze vicine all'origine, riduciamo la probabilità di incontrare valori nulli. In generale, comunque, i risultati che possono essere ottenuti tramite il filtraggio inverso sono di scarsa resa.[1]

Filtro di Wiener

Il metodo precedente non fa riferimento alla gestione del rumore. Questo invece considera sia le immagini che il rumore come variabili casuali e cerca di trovare un'approssimazione $\hat{g}$ dell'immagine originale g tale che l'errore quadratico medio tra di esse sia minimo. Questa misura dell'errore è data da:

$$e^2 = E\big[(g - \hat{g})^2\big] \tag{1.10}$$

dove $E[\cdot]$ è il *valore atteso* dell'argomento.

In questo caso si assume che il rumore e l'immagine non siano correlate, che l'uno o l'altra abbia media nulla e che i livelli di intensità nella stima siano una funzione lineare dei livelli nell'immagine degradata. Basandosi su queste condizioni, il minimo della funzione di errore (1.10) è dato, nel dominio della frequenza, dall'espressione:

$$\begin{aligned}\hat{G}(u,v) &= \left[\frac{H^*(u,v)S_g(u,v)}{S_g(u,v)|H(u,v)|^2 + S_n(u,v)}\right] F(u,v) \\ &= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_n(u,v)/S_g(u,v)}\right] F(u,v) \\ &= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_n(u,v)/S_g(u,v)}\right] F(u,v)\end{aligned}$$

dove

- $H(u,v)$ = *funzione di degrado*
- $H^*(u,v)$ = *coniugata complessa* di $H(u,v)$
- $|H(u,v)|^2 = H^*(u,v)H(u,v)$
- $S_n(u,v) = |S(u,v)|^2$ = *spettro di potenza* del rumore [4]
- $S_g(u,v) = |G(u,v)|^2$ = *spettro di potenza* dell'immagine non degradata.

Si fa uso del fatto che il prodotto di una quantità complessa con la sua coniugata è uguale alla magnitudo della quantità complessa al quadrato. Questo risultato è noto come *filtro di Wiener*[3] che per primo sviluppò tale concetto.

Dalla prima riga di (1.3), si può notare che il *filtro di Wiener* non presenta lo stesso problema del *filtro inverso* con gli zeri nella funzione di degrado, a meno che l'intero denominatore non sia zero per gli stessi valori di u e v.

[4] Il termine $|N(u,v)|$ viene anche detto *autocorrelazione del rumore*. Questo termine deriva dal teorema di correlazione. Stesse considerazioni possono essere fatte per $|G(u,v)|$ che rappresenta l'*autocorrelazione dell'immagine*.

Quando si ha a che fare con *rumore bianco spettrale* (*white noise*), lo spettro $|N(u,v)|^2$ è una costante e ciò semplifica molto le cose. Tuttavia, lo spettro di potenza dell'immagine non degradata è raramente noto: in tutti gli altri casi, il metodo consiste nell'approssimare l'equazione (**??**) tramite l'espressione:

$$\hat{G}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right] F(u,v) \tag{1.11}$$

dove K è una costante specificata che viene aggiunta a tutti i termini di $|H(u,v)|$.

Algoritmo Lucy-Richardson

I metodi di restaurazione basati sulla trasformata di Fourier danno buoni risultati solamente se la quantità di rumore nelle immagini degradate è moderato o nullo. All'aumentare di tale valore, tali metodi falliscono nel loro intento.

Quello di *Lucy-Richardson* è un algoritmo iterativo basato su una *stima Bayesiana*. Tale metodo rappresenta uno degli algoritmi di *deblurring* più utilizzati proprio perché produce delle buone soluzioni con ogni tipo di rumore dell'immagine. Inoltre, non richiede alcuna informazione sull'immagine latente.

I pixel dell'immagine osservata possono essere rappresentati tramite la PSF e l'immagine latente come:

$$f_i = \sum_j p_{ij} g_j \tag{1.12}$$

dove p_{ij} rappresenta la PSF, g_j è il valore del pixel in posizione j nell'immagine latente e d_i è il valore osservato nel pixel di posizione i.

Questo metodo assume che g_j segua una *distribuzione di Poisson.*[5] L'idea di base è quella di calcolare la g_j con probabilità più alta conoscendo f_i e p_{ij}. Quindi, tramite il *teorema di Bayes*, si vuole calcolare:

$$P(g_j|f_i) = \frac{P(f_i|g_j)P(g_j)}{\sum_k P(f_i|g_k)P(g_k)} \tag{1.13}$$

Si può inoltre dire che:

$$P(g_j) = \sum_i P(g_j f_i) = \sum_i P(g_j|f_i)P(f_i)$$

Ora, sostituendo l'equazione (1.13) in quella appena citata si ottiene:

$$P(g_j) = \sum_i \frac{P(f_i|g_j)P(g_j)P(f_i)}{\sum_k P(f_i|g_k)P(g_k)} \tag{1.14}$$

Nella formulazione pratica, la formula (1.14) diventa:

$$P_{t+1}(g_j) = P_t(g_j) \sum_i \frac{P(f_i|g_j)P(f_i)}{\sum_k P(f_i|g_k)P_t(g_k)} \tag{1.15}$$

Questo risultato è una procedura iterativa per il calcolo di $P(g_j)$ e come primo passo, $P_0(g_j)$ viene supposto con una *distribuzione di Poisson.*

L'equazione (1.15) può essere anche riscritta come:

$$g_j^{(t+1)} = g_g^{(t)} \sum_i \frac{f_i}{c_i} p_{ij} \tag{1.16}$$

dove

$$c_i = \sum_j p_{ij} u_j^{(t)}$$

[5]La *distribuzione di Poisson* è, $\forall n \in \mathcal{N}$ $P(n) = \frac{\lambda}{n!} e^{-\lambda}$ dove λ è il numero medio di eventi per intervalli di tempo e n è il numero di eventi per intervalli di tempo di cui si vuole calcolare la probabilità.

È stato empiricamente dimostrato che se questa iterazione converge, lo fa alla soluzione con *likelihood* massima per g_j.[4][5]

Infine, l'equazione (1.16) può essere scritta più generalmente in termini di *convoluzione*:

$$g^{(t+1)} = g^{(t)} \left(\frac{f}{g^{(t)} * p} * \hat{p} \right) \tag{1.17}$$

dove $\hat{p}$ è la *flipped* PSF tale che

$$\hat{p}_{nm} = p_{(i-n)(j-m)} \quad n \geq 0 \ , \ m \leq i, j$$

Capitolo 2

Deblurring nel dominio sparso

Come già accennato nel capitolo precedente, un'immagine degradata da *blur* viene rappresentata tramite la convoluzione:

$$f = g * p + n \tag{2.1}$$

tra l'immagine originale g e la PSF p con l'aggiunta di un rumore additivo n. Inoltre, una rappresentazione analoga può essere

$$f = Hp + n$$

dove H è una matrice che rappresenta l'operatore di convoluzione.

Per la risoluzione della deconvoluzione *blind*, quando cioè non siamo a conoscenza della funzione di degradazione p, esistono molti metodi. Possiamo suddividere tale problema in due categorie, in base a come viene identificato il *blur*:

- **Identificazione a *priori***

  In questo caso, prima si determina la PSF che, successivamente, viene utilizzata per stimare l'immagine originale tramite un algoritmo di *deblurring non-blind*. Per stimare la PSF si può assumere una forma

parametrica precisa di quest'ultima, come per esempio un *blur* lineare, in modo tale da riuscire ad identificarla solo tramite pochi parametri.

- **Identificazione congiunta**

  La maggior parte dei metodi esistenti ricade, comunque, in questo secondo tipo di stima, dove l'immagine e la PSF dovrebbero, teoricamente, essere determinate simultaneamente. Nella pratica tuttavia, molti di questi metodi usano un approccio alternato per stimare g e p invece che cercare una soluzione simultanea. Generalmente, anche in questi casi, vengono utilizzate delle conoscenze a priori (*prior*) sull'immagine e sul *kernel*.

I metodi di *deblurring blind* possono essere anche classificati in base al numero di immagini che si hanno a disposizione. Ovviamente, quando è possibile fare riferimento a più immagini di una stessa scena, la stima della PSF e, quindi, la risoluzione della deconvoluzione, risulta più semplice. Il problema è invece più complesso quando abbiamo a disposizione un'unica immagine degradata della scena.

In questo capitolo verrà introdurremo il concetto di *rappresentazione sparsa* che verrà poi applicato al metodo di deconvoluzione basato sulla formulazione di un problema di minimizzazione per la stima dell'immagine latente.

2.1 Rappresentazione sparsa

In alcune applicazioni, l'immagine può essere rappresentata tramite una *rappresentazione sparsa*, con rispetto ad una base B. In questo caso, x è definito come segue:

$$x = B\alpha \tag{2.2}$$

dove

- B rappresenta una matrice $n_p \times K$ le cui colonne sono *funzioni di base* (*basis functions*)
- α è un vettore di coefficienti

In alcuni casi B è *sovrabbondante* (*overcomplete,*) cioè $K > n_p$.

La matrice delle basi B può essere costruita da *basi standard*, come *Wavelet* o *Discrete Cosine Transform* (DCT). In altri casi è possibile costruire B tramite l'*apprendimento* (*learning*) partendo da un insieme di dati (in questo caso B è chiamato *dizionario*).

2.1.1 Costruzione del dizionario

Una criticità della rappresentazione sparsa è la scelta della matrice delle basi B. In questa sezione verranno illustrate le due tecniche principali: la prima basata sull'*apprendimento automatico* (*dictionary learning*), l'altra su un metodo analitico, tramite l'utilizzo dei *coefficienti Wavelet*.

Learning del dizionario

Il metodo dell'apprendimento di un *dizionario sparso* (*sparse dictionary learning*) è una texnica di *feature learning* che ha come obiettivo quello di trovare una rappresentazione sparsa di un dato in ingresso, espressa come come combinazione lineare di elementi di una base. Questi elementi vengono definiti *atomi* e costituiscono un *dizionario*.

Agli atomi di un dizionario non è richiesta la proprietà di *ortogonalità* e possono rappresentare un dizionario *overcomplete*: queste caratteristiche permettono di avere degli atomi ridondanti simili che ammettono più rappresentazioni dello stesso segnale, ma contribuiscono anche ad un miglioramento della sparsità e a una rappresentazione più flessibile.

Uno dei principi base del *dictionary learning* è che viene inferito dai dati in ingresso: in questo caso, il dizionario così costruito, può migliorare significativamente la sparsità.

Dato un dataset in ingresso $X = [x_1, \ldots, x_K]$ con $x_i \in \mathcal{R}^d$, l'obiettivo è quello di determinare un dizionario $D \in \mathcal{R}^{d \times n}$ e una rappresentazione $R = [r_1, \ldots, r_K]$ con $r_i \in \mathcal{R}^n$ tale che $\|X - DR\|_F^2$[1] sia minima e r_i sia abbastanza sparso. Ciò può essere formulato come il seguente problema di minimizzazione:

$$\arg\min_{D,r_i} \sum_{i=1}^{k} \|x_i - Dr_i\|_F^2 + \lambda \|r_i\|_0 \tag{2.3}$$

con il vincolo

$$C = \{D \in \mathcal{R}^{d \times n} : \|d_i\|_2 \leq 1 \quad \forall i = 1, \ldots, m\}$$

tale che gli atomi non raggiungano valori troppo alti per valori arbitrariamente bassi di α_i, ma comunque diversi da 0.

La risoluzione del problema citato sopra è *NP-hard* poiché la norma-L_0 rende la minimizzazione non convessa, si possono quindi utilizzare delle approssimazioni come MOD e K-SVD per la sua risoluzione. Nel primo caso l'idea è quella di risolvere la minimizzazione (2.3) soggetta ad un numero limitato di componenti diverse da zero:

$$\min_{D,R} \|X - DR\|_F^2 \quad \text{tale che} \quad \forall i \quad \|r_i\|_0 \leq T \tag{2.4}$$

Questo metodo determina prima la matrice R ,poi aggiorna il dizionario tramite la soluzione analitica del problema $D = XR^+$, dove R^+ indica la *pseudoinversa di Moore-Penrose*. Dopo tale aggiornamento D è normalizzata e si ripete il processo fino a che non converge.

[1] $\|X\|_F$ è la *norma di Frobenius* calcolata come $\|X\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} |a_{ij}|^2}$

L'algoritmo K-SVD effettua la *singular value decomposition* (SVD) aggiornando gli atomi del dizionario uno ad uno. Anche in questo caso il problema è definito come:

$$\min_{D,R} \|X - DR\|_F^2 \quad \text{tale che} \quad \forall i \|\alpha_i\|_0 \leq T_0 \tag{2.5}$$

Questo algoritmo prima fissa il dizionario, poi determina la migliore R sotto determinati vincoli (utilizzando OMP) e in seguito aggiorna gli atomi del dizionario D nel seguente modo:

$$\|X - DR\|_F^2 = \left| X - \sum_{i=1}^{K} d_i x_T^i \right|_F^2 = \|E_k - d_k x_T^k\|_F^2$$

I passi successivi consistono nell'*approssimazione di rango-1* della matrice dei residui E_k, nell'aggiornamento di d_k e nell'imposizione della sparsità di x_k dopo l'aggiornamento. Questi metodi vengono utilizzati e adattati al problema del *deblurring* in [6] e [7].

Trasformata Wavelet

Un metodo per la costruzione della matrice delle basi B per via analitica è quello che utilizza la *trasformata Wavelet*.

Questa è molto utilizzata nel campo dell'*image processing*, poiché, oltre alle informazioni di un segnale in frequenza, mantiene anche il contenuto informativo nel tempo di tale segnale. Permette inoltre una rappresentazione su più scale e la sua realizzazione si ha tramite un banco di filtri.

In questo senso, consideriamo un'immagine $f(x, y)$ di dimensioni $M \times N$. La *trasformata Wavelet bidimensionale* (DWT 2-D) (vedi Appendice A) di $f(x, y)$ può essere implementata tramite filtri digitali e sottocampionatori. Con *funzioni Wavelet* e di *scaling* bidimensionali separabili, basta eseguire la *trasformata Wavelet monodimensionale* (DWT 1-D) delle righe di $f(x, y)$,

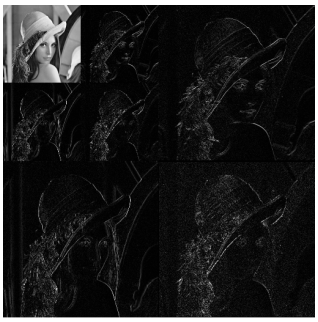


Figura 2.1: Immagine *Lena* decomposta con due livelli Wavelet

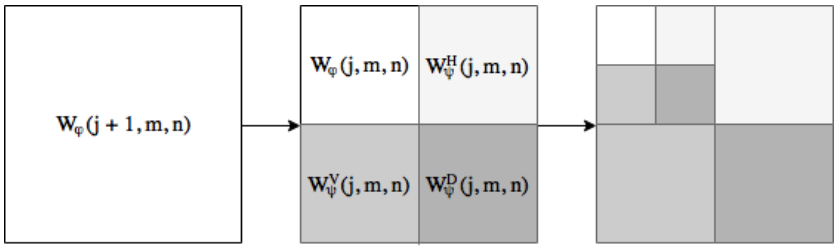


Figura 2.2: Decomposizione risultante

seguita dalla DWT 1-D delle colonne risultanti. Tale processo è mostrato in figura (2.2).

Nel caso bidimensionale, si ottengono tre insiemi di *coefficienti di dettaglio*: orizzontali, verticali e diagonali.

Il banco di filtri della figura (2.3) può essere iterato per produrre una trasformata a P livelli. L'immagine $f(x, y)$ è utilizzata come input $W_\phi(J, m, n)$. Mediante il processo di convoluzione con $h_\phi(-n)$ e $h_\psi(-n)$ e sottocampionando le sue colonne, si ottengono due sottoimmagini le cui risoluzioni orizzontali vengono ridotte di un fattore 2.

La componente *high-pass* (o di *dettaglio*) caratterizza le informazioni ad alta frequenza delle immagini con orientamento verticale; la componente di approssimazione *low-pass* contiene le informazioni verticali a bassa frequenza. Entrambe le sottoimmagini vengono poi filtrate per colonne e sottocam-

pionate per ottenere quattro sottoimmagini di output dalle dimensioni di un quarto (2.2): W_ϕ, W_ψ^H, W_ψ^V e W_ψ^D che rappresentano rispettivamente le basse frequenze, i dettagli sulle righe, i dettagli sulle colonne e quelli sulle diagonali.

Due iterazioni del processo di filtraggio producono la decomposizione mostrata nella parte destra dell'immagine (2.2).

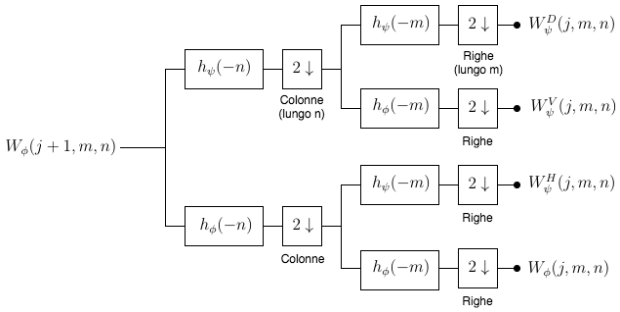


Figura 2.3: Banco dei filtri di analisi

La figura (2.4) mostra il banco di filtri di sintesi che inverte il processo descritto sopra. Le sottoimmagini di approssimazione j a quattro scale e quella di dettaglio vengono sottocampionate e convolute con due filtri unidimensionali, uno operante sulle colonne delle sottoimmagini, l'altro sulle righe.

La somma dei risultati porta all'approssimazione di scala $j+1$ e il processo viene ripetuto finché l'immagine originale non viene ricostruita.

Un esempio dell'applicazione di tale trasformata si ha in figura (2.1). Per una teoria più dettagliata su tale trasformata, si rimanda all'Appendice A per una teoria più dettagliata su tale trasformata.[8]

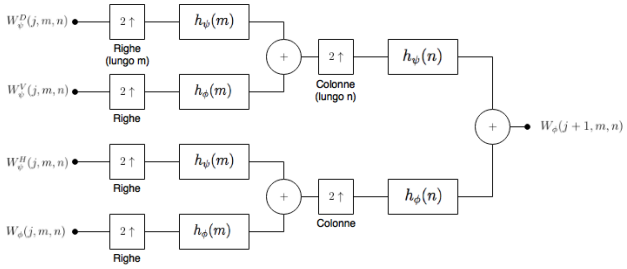


Figura 2.4: Banco dei filtri di sintesi

2.2 Il problema della deconvoluzione

Uno dei metodi per determinare l'immagine latente è quello di esprimere la deconvoluzione come un problema di minimizzazione, chiamato *Linear Least Square*, espresso come:

$$\hat{g} = \arg\min_g \|f - Hg\|_2^2 \tag{2.6}$$

Poiché questo problema di deconvoluzione è mal-posto, e la soluzione è generalmente non unica, tale approccio è troppo semplificato.

Per trovare una migliore soluzione di (2.1) possono essere utilizzate delle conoscenze a *priori* (*prior knowledge*) sulle immagini per *regolarizzare* il problema.

Uno dei modelli più comuni è la *regolarizzazione di Tikhonov*. Allo scopo di trovare una particolare soluzione con le proprietà desiderate, all'equazione (2.6) viene sommato un *termine di regolarizzazione* tale che:

$$\hat{g} = \arg\min_g \|f - Hg\|_2^2 + \|\Gamma g\|_2 \tag{2.7}$$

per qualche scelta opportuna della *matrice di Tikhonov* Γ.

In molti casi, la scelta cade sulla *matrice identità*, $\Gamma = I$, preferendo soluzioni con norma più piccola. In altri casi, operatori passa alto, come per

esempio un operatore discreto di Fourier opportunamente pesato, possono essere impiegati per rafforzare il carattere *smooth* di g quando è ritenuto principalmente continuo.

Questa regolarizzazione migliora il condizionamento del problema, rendendo possibile una soluzione di tipo numerico. Tale soluzione in forma chiusa è data da:

$$\hat{g} = (H^T H + \Gamma^T \Gamma)^{-1} H^T f$$

Un altro tipo di regolarizzazione tipicamente usato è il *Total Variation*, con cui la minimizzazione (2.6) diventa:

$$\hat{g} = \arg\min_g \|f - Hg\|_2^2 + \lambda \|\nabla g\|_1 \tag{2.8}$$

dove ∇g è la norma-L_1 della derivata del primo ordine di g e λ è una costante. Poiché questa regolarizzazione favorisce le strutture costanti a tratti delle immagini, tende a eliminarne i dettagli.

In questo lavoro, il *prior* utilizzato per regolarizzare la minimizzazione (2.6) è basato sulla rappresentazione sparsa dell'immagine, come definito nella sezione precedente.

I metodi basati sulla sparsità si basano sull'assunto che solamente pochi valori del vettore dei coefficienti siano diversi da 0. A questo proposito, la norma-L_0[2] conta il numero di elementi diversi da 0.

Consideriamo il problema $y = Hx + n$, dove $y \in \mathcal{R}^{M \times N}$ è l'immagine *blurred*, $x \in \mathcal{R}^{MN}$ è la rappresentazione lessicografica dell'immagine latente, $H \in \mathcal{R}^{MN \times MN}$ è la matrice che rappresenta la convoluzione ed $n \in \mathcal{R}^{M \times N}$ è il rumore additivo.

[2]La norma $\|x\|_0 = \sum_i 1_{\{x_i \neq 0\}}$

L'immagine latente x può quindi essere rappresentata tramite una matrice di basi, B, e un vettore dei coefficient z come:

$$x = Bz$$

Il problema di determinare l'immagine latente, secondo le assunzioni fatte sopra, può essere fatto in due modi diversi:

- **Metodo della sintesi**

  Si risolve il seguente problema di minimizzazione basato sulla norma-L_0

  $$\hat{z} = \arg\min_z \|y - HBz\|_2^2 + \lambda\|z\|_0 \tag{2.9}$$

  dove la matrice B è chiamata *matrice di sintesi*. Una volta ottenuto z, x può essere stimato come $\hat{x} = B\hat{z}$

- **Metodo dell'analisi**

  In questo caso si risolve il problema

  $$\hat{x} = \arg\min_x \|y - Hx\|_2^2 + \lambda\|B^T x\|_0 \tag{2.10}$$

  dove B è detta *matrice di analisi*.

Quando B è invertibile, il metodo della sintesi e dell'analisi sono identici.

Poiché la norma $\|\cdot\|_0$ è non convessa e non differenziabile, i problemi di minimizzazione associati sono problemi di ricerca *NP-hard*. L'idea quindi è quella di utilizzare un *rilassamento convesso* di tale norma, utilizzando per esempio la norma-L_1[3]: quest'ultima rappresenta, infatti, la funzione convessa più vicina alla $\|\cdot\|_0$ e per questo motivo è molto utilizzata.

Il rilassamento del problema (2.9) diventa quindi:

$$\hat{z} = \arg\min_z \|y - HBz\|_2^2 + \lambda\|z\|_1 \tag{2.11}$$

Questo problema è conosciuto come *Regularized Least Square* del quale, nel capitolo successivo, verrà approfondito un metodo per la sua risoluzione.

[3]La norma $\|x\|_1 = \sum_i |x_i|$

Capitolo 3

Realizzazione

Dopo aver introdotto il concetto di sparsità e di come può essere applicata al problema della ricostruzione delle immagini, andiamo, in questo capitolo, ad adattarlo a immagini corrotte da *motion blur* lineare.

Nella sezione (3.1) viene mostrato il metodo *non-blind*, cioè viene effettuata una stima dell'immagine latente conoscendo, a priori, la PSF.

Nella sezione (3.2) viene definito, invece, un metodo per determinare la PSF basata sull'uso della *trasformata Radon*. Dopo averla determinata, viene utilizzato il metodo implementato per il *deblurring non-blind* per ricostruire l'immagiine originale.

3.1 *Non-blind deblurring*

Consideriamo un'immagine degradata $f \in \mathcal{R}^{M \times N}$, definita come:

$$f = g * p + n \tag{3.1}$$

dove $g \in \mathcal{R}^{M \times N}$, $p \in \mathcal{R}^{M \times N}$ è il *blur kernel* e $n \in \mathcal{R}^{M \times N}$ il rumore.

La tecnica di *deblurring non-blind* utilizzata è basata sul metodo della *regolarizzazione*. Si tratta di un problema di deconvoluzione tale che, dato il

blur kernel p, si determina l'immagine latente g risolvendo la minimizzazione:

$$\hat{g} = \arg\min_{g} 1/2\|p * g - f\|_2^2 + \lambda\Theta(g) \tag{3.2}$$

dove $\Theta(g)$ rappresenta il *termine di regolarizzazione* e λ è il corrispondente *parametro di regolarizzazione*.

Per risolvere l'equazione (3.2) vengono utilizzate delle conoscenze a priori sulla sparsità dell'immagine g in un sistema *Wavelet*, come definito nella sezione (2.2), per regolarizzare la minimizzazione. Tale termine, basato sulla rappresentazione sparsa è:

$$\Theta(g) = \|W_d g\|_1 \tag{3.3}$$

dove g è la forma vettorizzata dell'immagine latente e W è la matrice delle basi, costruita tramite la trasformata Wavelet.

La matrice W_d rappresenta l'operatore di *analisi*. Definendo tale operatore, si ha che moltiplicando la matrice W_d con g, si ottiene la trasformata Wavelet di quest'ultima. Si definisce inoltre l'operatore di *sintesi* tale che:

$$\hat{g} = W_r(W_d g)$$

.

Il problema della deconvoluzione si può quindi riscrivere come:

$$\hat{g} = \arg\min_{g} 1/2\|p * W_r(W_d g) - f\|_2^2 + \lambda\|(W_d g)\|_1 \tag{3.4}$$

definendo $u = W_d g$ si ha che (3.4) diventa:

$$\hat{u} = \arg\min_{u} 1/2\|p * W_r u - f\|_2^2 + \lambda\|u\|_1 \tag{3.5}$$

Si può quindi ricostruire l'immagine latente come

$$\hat{g} = W_r \hat{u}$$

.

Per risolvere l'equazione (3.5) viene, inoltre, introdotta una rappresentazione matriciale dell'operatore di convoluzione, la cui costruzione verrà definita in (3.1).

Possiamo quindi riscrivere l'operazione di convoluzione tra l'immagine e la PSF come un prodotto matrice-vettore, cioè

$$p * g = Hg$$

Il problema diventa quindi quello di risolvere:

$$\hat{u} = \arg\min_u 1/2\|HW_r u - f\|_2^2 + \lambda\|u\|_1 \tag{3.6}$$

Definendo $A = HW_r$, (3.6) diventa un problema di minimizzazione *regularized least square* (LS), cioè

$$\hat{u} = \arg\min_u 1/2\|Au - f\|_2^2 + \lambda\|u\|_1 \tag{3.7}$$

con $\hat{g} = W_r\hat{u}$.

Per risolvere l'equazione (3.7) sono stati utilizzati i pacchetti GPSR [9] e L1_Ls [10], implementati in MATLAB [11] e illustrati nella sezione (3.1).

Matricizzazione

In questa tesi è stato sfruttato il fatto che un operatore lineare può essere rappresentato in forma matriciale. In particolare, in questa sezione, facciamo vedere come è possibile costruire la *ìmatrice di filtraggio*, utilizzata per effettuare la convoluzione tra l'immagine e la PSF, e quella che rappresenta il *downsampling* utilizzata per la costruzione delle basi Wavelet.

Matrice di filtraggio

Consideriamo un'immagine $X \in \mathcal{R}^{M\times N}$ e un filtro $P \in \mathcal{R}^{L\times K}$. La *convolu-*

zione lineare 2D tra X e P è definita come:

$$Y(m,n) = X(m,n) * P(m,n) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X(i,j)P(m-i,n-j) \quad (3.8)$$

Il risultato è, quindi, una matrice di dimensioni $M \times N$.

La matrice di filtraggio H è costruita in modo tale che:

$$X * P = HP \quad (3.9)$$

Inoltre, poiché l'operazione di convoluzione è *separabile*, cioè può essere eseguita prima per colonne e poi per righe, possiamo riscrivere la (3.9) come:

$$HP = H_r H_c p \quad (3.10)$$

dove H_r rappresenta il *filtraggio per righe* mentre H_c quello *per colonne*. Vediamo come è possibile costruire tali matrici.

Consideriamo il vettore $x \in \mathcal{R}^{MN\times 1}$ dell'immagine, ottenuto incolonnando la matrice X, e il vettore $p = (p_0, p_1, \ldots, p_L K)$ del filtro, ottenuto incolonnando la matrice P dopo averne invertito righe e colonne.

- **Matrice di filtraggio per colonne**

  La matrice $B \in \mathcal{R}^{M\times M}$ costruita come:

$$\begin{vmatrix} p_0 & 0 & 0 & \cdots & 0 \\ p_1 & p_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ p_{LK} & p_{LK-1} & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & p_2 & p_1 & p_0 \end{vmatrix} \quad (3.11)$$

  è la matrice di filtraggio che agisce sulla prima colonna di X.

  Per far si che questa agisca su tutte le colonne, viene eseguito il *prodotto di Kronecker*[1] tra la la *matrice identità* $I \in \mathcal{R}^{N\times N}$ e B.

[1]Il *prodotto di Kronecker* tra due matrici, $X \otimes Y$, esegue il prodotto di ogni elemento di X per la matrice Y

Il risultato è la matrice di filtraggio per colonne $H_c \in \mathcal{R}^{MN \times MN}$ che presenta la seguente struttura, definita *matrice di Toeplix a blocchi*:

$$\begin{vmatrix} B & 0 & 0 & \cdots & 0 \\ 0 & B & 0 & \cdots & 0 \\ 0 & 0 & B & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & B \end{vmatrix} \tag{3.12}$$

- **Matrice di filtraggio per righe**

  Come nel caso precedente, si definisce la matrice di filtraggio per righe partendo dalla matrice $B \in \mathcal{R}^{N \times N}$ definita come nel caso precedente. Affinchè questa agisca su tutte le righe, viene eseguito il prodotto di Kronecker tra B e la matrice identità $I \in \mathcal{R}^{M \times M}$.

  Il risultato è la matrice di convoluzione delle righe $H_r \in \mathcal{R}^{MN \times MN}$ seguente:

$$\begin{vmatrix} p_0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \cdots \\ 0 & p_0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & \cdots \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & 0 & \cdots \\ 0 & 0 & \cdots & p_0 & 0 & \cdots & \cdots & 0 & \cdots \\ p_1 & 0 & \cdots & \cdots & p_0 & 0 & \cdots & 0 & \cdots \\ 0 & p_1 & 0 & \cdots & 0 & p_0 & \ddots & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & p_1 & 0 & \cdots & 0 & p_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{vmatrix} \tag{3.13}$$

Infine, è possibile eseguire la convoluzione tra il filtro p e l'immagine x tramite il prodotto matrice-vettore:

$$H_c H p x$$

Downsampling

Come definito nella sezione (2.1.1), la trasformata Wavelet viene eseguita, utilizzando un banco di filtri, tramite delle convoluzioni. In particolare, le immagini (e le sottoimmagini), vengono *sottocampionate* tramite un filtro, detto di *downsampling*.

Anche questa operazione è lineare e può, quindi, essere rappresentata come un prodotto matrice-vettore. Vediamo adesso come costruire tale matrice di filtraggio.

Consideriamo la matrice $X \in \mathcal{R}^{M \times N}$. Sottocampionare significa ridurre il numero di campioni utilizzati per rappresentare un segnale. Nel caso delle immagini questo si ha eliminando le componenti ad alta frequenza, mantenendo quelle a bassa.

Sottocampionare di un *fattore 2* un'immagine, consiste nel mantenere le colonne e righe dispari dell'immagine, eliminando il resto dei campioni. Questo porta, con riferimento ad X, a ottenere una matrice

$$X_d \in \mathcal{R}^{\frac{M}{2}\frac{N}{2}}$$

.

Anche in questo caso, si può considerare il filtraggio di *downsampling* separabile. Consideriamo, per semplicità, $M = N$, cioè un'immagine quadrata, la cui forma vettoriale è $x \in \mathcal{R}^{M^2 \times 1}$.

Le matrici di *downsampling*, per un sottocampionamento di fattore 2, sono così ottenute:

- **Matrice di *downsamplig* delle colonne**

  Un *pattern* ricorrente in questa matrice è costituito da vettore colonna, definito come $b = [1, 0]$, che viene quindi utilizzato in entrambe le costruzioni delle matrici di *downsampling*.

Il primo passo è costruire la matrice $B \in \mathcal{R}^{\frac{M}{2} \times M}$, definita come il prodotto di Kronecker tra la matrice identità $I \in \mathcal{R}^{\frac{M}{2} \times \frac{M}{2}}$ e b.
Si ottiene così la matrice $E \in \mathcal{R}^{\frac{M}{2} \times M}$, costruita come:

$$\begin{vmatrix} b & 0 & \cdots & 0 \\ 0 & b & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & b \end{vmatrix} \tag{3.14}$$

Affinchè il filtraggio avvenga su tutte le colonne, si effettua il prodotto di Kronecker tra E e la matrice identità di dimensioni $M \times M$. Si ottiene in questo modo la matrice di *downsampling* delle colonne $E_c \in \mathcal{R}^{\frac{M}{2} M \times M^2}$.

- **Matrice di *downsampling* delle righe** Consideriamo il vettore b, come definito nel caso precedente e di aver effettuato l'operazione di sottocampionamento delle colonne su X tale che la sia adesso costituita da M righe e $\frac{M}{2}$ colonne.
  Eseguendo il prodotto di Kronecker tra la matrice identità $I \in \mathcal{R}^{\frac{M}{2} \times \frac{M}{2}}$ e il vettore b, si ottiene la matrice $E \in \mathcal{R}^{\frac{M}{2} \times M}$ definita come nel caso precedente.
  Per ottenere la matrice di *downsampling* delle righe, si esegue il prodotto di Kronecker tra la matrice identità $I \in \mathcal{R}^{\frac{M}{2} \times \frac{M}{2}}$ e E ottenendo la matrice a blocchi $E_r \in \mathcal{R}^{\frac{M^2}{4} \times \frac{M}{2} M}$

$$\begin{vmatrix} E & 0 & \cdots & 0 \\ 0 & E & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & E \end{vmatrix} \tag{3.15}$$

In questo modo si ottiene il sottocampionamento (di fattore 2) eseguendo

il prodotto matrice-vettore

$$E_r E_c x$$

.

Risolutori di LS

Consideriamo il problema di ottimizzazione convesso non vincolato:

$$\min_x \frac{1}{2}\|Ax - y\|_2^2 + \tau\|x\|_1 \tag{3.16}$$

dove $x \in \mathcal{R}^n$, $y \in \mathcal{R}^k$, A è una matrice $k \times n$ e τ è un parametro non negativo. La presenza della norma-L_1 fa si che valori piccoli valori in x diventino esattamente 0 e produce una soluzione sparsa.

Il problema (3.16) è strettamente legato ai seguenti problemi di ottimizzazione convessi e vincolati:

$$\min_x \|x\|_1 \quad \text{tale che} \quad \|y - Ax\|_2^2 \leq \epsilon \tag{3.17}$$

e

$$\min_x \|y - Ax\|_2^2 \quad \text{tale che} \quad \|x\|_1 \leq t \tag{3.18}$$

Il problema (3.17) è un *Quadratically Constrained Linear Program* (QCLP) mentre (3.18) è un *Quadratic Program* (QP).

Può essere dimostrato che una soluzione di (3.17) può essere $x = 0$ e un minimizzatore di (3.16), per $\tau \geq 0$. Allo stesso modo una soluzione di (3.18) per ogni $t \geq 0$ è un minimizzatore di (3.16) per qualche $\tau \geq 0$.

In questa sezione, vengono brevemente descritti i due *solver*, implementati in MATLAB, utilizzati per la risoluzione del problema (3.16).

- **GPSR**

  Una criticità della minimizzazione definita in (3.16) è dovuta dalla dimensione della matrice A. I problemi principali sono due:

- spesso non può essere memorizzata esplicitamente a causa delle sue dimensioni
- è molto costoso accedere a porzioni di A e A^TA

Il pacchetto GPSR (*Gradient Projection for Sparse Reconstruction*) richiede solo prodotti matrice-vettore che coinvolgono A e A^T invece che un accesso esplicito ad A.

Il metodo di risoluzione è essenzialmente un algoritmo *Gradient Projection* (GP) applicato ad una formulazione *Quadratic Programming* di (3.16), dove il cammino ad ogni iterazione è ottenuto proiettando la direzione negativa del gradiente in un *set ammissibile*.

Per risolvere il problema (3.16), questo viene definito come un QP. Per farlo, viene suddivisa la variabile x in parte positiva e negativa, nel seguente modo:

$$x = u - v \quad u \geq 0,\, v \geq 0$$

Inoltre si ha che :

$$\|x\|_1 = 1_n^T u + 1_n^T v$$

dove 1_n è il vettore colonna composto da n uno.

Si può notare, quindi, che (3.16) può essere riscritto come un *Bound-Constrained Quadratic Program* (BCQP):

$$\min_{u,v} \frac{1}{2}\|y - A(u - gv)\|_2^2 + \tau 1_n^T u + \tau 1_n^T v \quad \text{tale che} \quad u, v \geq 0 \tag{3.19}$$

La forma standard del BCQP è la seguente:

$$\min_x c^T z + \frac{1}{2} z^T B z \equiv F(z) \quad \text{tale che} \quad z \geq 0 \tag{3.20}$$

dove

$$z = \begin{bmatrix} u \\ v \end{bmatrix}, \quad b = A^T y, \quad c = \tau 1_{2n} + \begin{bmatrix} -b \\ b \end{bmatrix}$$

e

$$B = \begin{bmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{bmatrix}$$

La tecnica utilizzata per risolvere (3.20) è una tecnica GP. Il passaggio dall'iterazione $z^{(k)}$ alla $z^{(k+1)}$ è la seguente:

- viene scelto un parametro scalare $\alpha^{(k)} \geq 0$ e si determina

$$w^{(k)} = (z^{(k)} - \alpha^{(k)} \nabla F(z^{(k)})_+$$

- si considera un secondo scalare $\lambda^{(k)} \in [0, 1]$ e si calcola

$$z^{(k+1)} = z^{(k)} + \lambda^{(k)}(w^{(k)} - z^{(k)})$$

Per la scelta dei valori di $\lambda^{(k)}$ e $\alpha^{(k)}$ e sulla convergenza di tale metodo si fa riferimento all'articolo [9].

Questo metodo definisce vari criteri di arresto. Quello utilizzato considera la norma della differenza tra due stime successive, diviso la norma di una di queste e termina quando tale valore scende sotto una certa costante *tol*. Cioè

$$\frac{\|z^{(k)} - z^{(k+1)}}{\|z^{(k)}\|} \leq tol \tag{3.21}$$

Per altri criteri di terminazione e approfondimenti, consultare [9].

- **L1-Ls**

Questo pacchetto, utilizza una variante della formulazione (3.18) per risolvere il problema (3.16). Considera, cioè, un problema quadratico convesso, con vincoli di disuguaglianza lineari, cioè:

$$\min_x \|Ax - y\|_2^2 + \lambda \sum_{i=1}^{n} u_i$$

$$\text{tale che} \quad -u_i \leq x_i \leq u_i, \quad i = 1, \ldots, n$$

dove le variabili sono $x \in \mathcal{R}^n$ e $u \in \mathcal{R}^n$.

Consideriamo, adesso, la *barrier function* logaritmica del vincolo $-u_i \leq x_i \leq u_i$, definita come

$$\phi(x,u) = -\sum_{i=1}^{n} \log(u_i + x_i) - \sum_{i=1}^{n} \log(u_i - x_i) \tag{3.22}$$

Il cammino centrale consiste nell'unico minimizzatore $(x^*(t), y^*(t))$ della funzione convessa

$$\Phi_t(x,u) = t\|Ax - y\|_2^2 + t\sum_{i=1}^{n} \lambda u_i + \phi(x,y) \tag{3.23}$$

con $t = 0, \ldots, \infty$.

Con il metodo *Interior Point* si vanno a cercare solamente i punti interni allo *spazio ammissibile*, per una sequenza di valori t.

Per minimizzare $\Phi_t(x,y)$, definita in (3.23), si utilizza il *Metodo di Newton*, dove la direzione del cammino è definita risolvendo il *sistema di Newton*

$$H \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} = -g \tag{3.24}$$

dove $H = \nabla^2 \Phi_t(x,u) \in \mathcal{R}^{2n \times 2n}$ è l'*Hessiana* e $g = \nabla \Phi_t(x,u) \in \mathcal{R}^{2n}$ è il *gradiente* alla corrente iterazione (x,u).

Poiché risolvere il sistema di Newton (3.24) non è computazionalmente pratico, per determinare la direzione, si usa un'approssimazione della soluzione compiuta tramite il metodo *Preconditioned Conjugate Gradients* (PCG).[10]

3.1.1 Implementazione dell'algoritmo *non-blind*

In questa sezione viene data una breve descrizione dell'algoritmo implementato in MATLAB [11].

L'algoritmo prende in ingresso un'immagine non degradata e i parametri della PSF che vogliono essere applicati, cioè $LENGTH$, che rappresenta la lunghezza della PSF, e Θ che rappresenta l'angolo.

Viene quindi ricostruita la matrice della PSF tramite la funzione *fspecial* del pacchetto "Image Processing" di MATLAB che crea il *kernel* del filtro lineare che una volta applicato produce il *motion blur*. Questa matrice viene utilizzata dal metodo *matrix_filter2D*, che crea la matrice di filtraggio H, come definito nella sezione (3.1).

L'immagine blurred y si ottiene come il prodotto tra l'immagine in ingresso (in forma vettoriale) e la matrice H. Successivamente si crea la matrice di sintesi della trasformata *Wavelet* W_r, la quale viene moltiplicata con la matrice H determinando così la matrice A, utilizzata dal metodo *GPSR_BB* del pacchetto GPSR, o da L1_Ls, per risolvere il problema di minimizzazione.

Il risultato è un vettore x contenente i coefficienti della minimizzazione e si determina così l'immagine latente come $W_r x$ del quale viene fatto un *reshape* per riportare l'immagine alle sue dimensioni originali.

I parametri di significativa importanza per l'algoritmo, in particolare per il metodo di risoluzione del problema quadratico, sono:

- λ utilizzato per determinare il grado di sparsità della soluzione. Generalmente ad un valore alto viene associato un basso grado di sparsità.
- *tollerance* è il parametro utilizzato per l'arresto delle routine GPSR, come definito nella sezione (**??**), e L1_Ls.

Di seguito viene presentato lo pseudocodice per effettuare il *deblurring non-blind*.

Data: Immagine *blurred* y, parametri della PSF Θ e $LENGTH$, fattore di regolarizzazione λ

Result: Immagine *deblurred* X

1 Rendere y vettore colonna;

2 Creare la matrice di *blur* PSF tramite i suoi parametri;

3 Creare la matrice di filtraggio H utilizzando PSF;

4 Creare la matrice di ricostruzione *Wavelet* W_r;

5 $A = H * W_r$;

6 Risoluzione di $\|y - Ax\|_2^2 + \lambda\|x\|_1$ utilizzando la routine GPSR o L1_Ls;

7 $X = reshape(W_r x)$

Algorithm 1: Deblurring non-blind

3.2 *Blind deblurring*

La risoluzione del problema di *deblurring blind* è ancora più mal-posto. Infatti, oltre all'immagine latente, deve essere stimato anche il *kernel* del filtro che ha determinato la degradazione dell'immagine.

Per risolvere questo tipo di problema viene fatta l'ipotesi che la PSF rappresenti un *motion blur* lineare: in questo caso basta quindi determinare i parametri di tale filtro, cioè la *lunghezza*, che rappresenta la velocità dello spostamento, e l'*angolo*, cioè la sua direzione.

Dopo aver determinato tali parametri, viene utilizzata la routine di *deblurring non-blind* (1) per determinare l'immagine latente.

Nella prossima sezione viene definito come sono stati determinati i parametri della PSF.

3.2.1 Stima della PSF

Per effettuare una stima della PSF, si deve considerare, innanzitutto, che nel caso discreto la lunghezza del *blur* è caratterizzata dal numero di *pixel* della matrice che rappresenta la PSF ed è proporzionale alla velocità del movimento. Inoltre, l'angolo, può variare tra $0° - 180°$.

Una limitazione dovuta alla discretizzazione della PSF è data dall'impossibilità di distinguere due *blur kernel* che hanno angoli simili, soprattutto quando la lunghezza del *kernel* è piccola.

Consideriamo quindi il problema

$$f(x, y) = g(x, y)p(x, y) + n(x, y)$$

La sua trasformata di Fourier è

$$F(u, v) = G(u, v)P(u, v) + N(u, v) \tag{3.25}$$

dove F, G, P, N rappresentano rispettivamente le trasformate di Fourier dell'immagine *blurred*, di quella latente, della PSF e del rumore.

Come è comune nei problemi di deconvoluzione, non consideriamo l'apporto del rumore. Consideriamo adesso lo *spettro di potenza* di F su base logaritmica, definito da:

$$\log |F(u, v)| = \log |G(u, v)P(u, v)| \tag{3.26}$$

Si nota che il comportamento di $F(u, v)$ dipende essenzialmente dal prodotto $G(u, v)P(u, v)$. Inoltre $P(u, v)$ ha la struttura di un *sinc* che viene preservata anche per $F(u, v)$: in presenza di rumore, gli zeri del *sinc* diventano minimi locali.

Per determinare l'angolo e la lunghezza della PSF si fa uso della *trasformata Radon* che verrà introdotta nella prossima sezione.

Trasformata Radon

Definiamo la *trasformata Radon* di una funzione $\phi(x, y) \in \mathcal{R}^2$, all'angolo θ e distanza ρ dall'origine come:

$$R(\phi, \rho, \theta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \phi(x, y)\delta(\rho - x\cos\theta - y\sin\theta)dxdy \tag{3.27}$$

dove δ denota la *funzione di Dirac*. Equivalentemente

$$R(\phi, \rho, \theta) = \int_{-\infty}^{+\infty} \phi(\rho\cos\theta - s\sin\theta, \rho\sin\theta + s\cos\theta)dxdy \tag{3.28}$$

Nel caso discreto, l'equazione (3.27) diventa

$$R(\phi, \rho, \theta) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \phi(x, y)\delta(x\cos\theta + y\sin\theta - \rho) \tag{3.29}$$

Fissando θ e variando ρ, l'equazione (3.29) somma i pixel (x, y) lungo la retta definita tramite tali parametri. Incrementando i valori di ρ richiesti per coprire tutta l'immagine (con θ fissato) si ottiene una *proiezione*.[1]

Stima dell'angolo

Per determinare l'angolo della PSF prima di tutto, viene filtrata l'immagine con un una *finestra di Hann* (*Hann window*). Il suo utilizzo è dovuto al fatto che la trasformata di Fourier tratta i dati come se fossero infiniti ma l'immagine ha dimensioni finite. Questo crea delle alterazioni del contenuto in frequenza, causate dai bordi dell'immagine. Tramite il *windowing* vengono eliminati tali alterazioni.

La funzione utilizzata è quella di Hann, tale che:

$$w(n) = 0.5(1 - \cos\left(\frac{2\pi n}{C-1}\right)) \tag{3.30}$$

con $n = 0, \ldots, C-1$ dove C è la larghezza della finestra. Il risultato dell'utilizzo della finestra di Hann è mostrato in figura (3.3).

Figura 3.1: *Lena* blurred con $L = 20$, $\theta = 80$

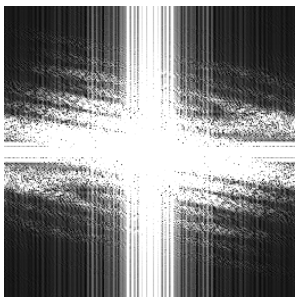


Figura 3.2: Spettro di potenza senza finestra di Hann

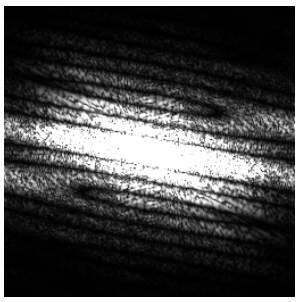


Figura 3.3: Spettro di potenza con finestra di Hann

Dopo aver calcolato la trasformata di Fourier dell'immagine finestrata, andiamo a studiare lo spettro del segnale, su scala logaritmica, cioè:

$$\log |1 + X(u, v)| \tag{3.31}$$

dove $X(u, v)$ è la trasformata di Fourier dell'immagine in ingresso. Nelle figure (3.2) e (3.3) è mostrato il contenuto in frequenza di (3.1).

Lo spettro presenta una certa struttura. Infatti, è costituito da fasce luminose che hanno una direzione ortogonale rispetto all'angolo del *motion blur*. Viene quindi applicata la trasformata Radon allo spettro $\log |X(u, v)|$ e il risultato è una matrice contenente, per ogni angolo (nelle colonne) $\theta = [0° - 180°]$, il valore della trasformata variando ρ (righe).

Inoltre si ha che la varianza di $R(\log|F(u,v)|,\rho,\theta)$, in funzione di θ, dipende solamente dall'orientamento del *blur kernel*.

La stima dell'angolo è quindi data da:

$$\hat{\theta} = \arg\max_{\theta} var\{R(\log|F(u,v),\rho,\theta)\} \tag{3.32}$$

dove $var\{\cdot\}$ è la varianza del set di valori ottenuti variando ρ. Nell'applicazione pratica, viene calcolata la varianza per ogni colonna della matrice che rappresenta la trasformata Radon e, del vettore risultante,ne viene determinato il massimo. L'indice di tale valore coincide con l'angolo stimato.[12]

Stima della lunghezza

Dopo aver stimato l'angolo $\hat{\theta}$, questo viene utilizzato per determinare la lunghezza della PSF.

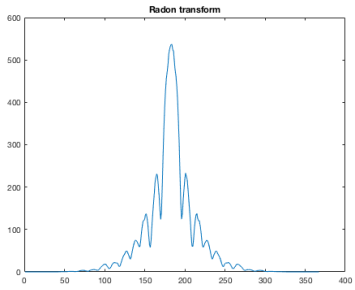


Figura 3.4: Trasformata Radon di (3.1)

Poiché la trasformata Radon calcolata nell'angolo stimato $\hat{\theta}$ (3.4), cioè

$$R(\log|F(u,v),\rho,\hat{\theta})$$

, mantiene ancora la struttura del *sinc*, si può notare che i punti di minimo rappresentano le line scure che sono circa alla solita distanza l'una dall'altra. Si ha quindi che la lunghezza stimata corrisponde alla distanza media fra i punti di minimo di $R(\log|F(u,v),\rho,\hat{\theta})$.[13]

3.2.2 Implementazione dell'algoritmo *blind*

L'ingresso dell'algoritmo *blind* è l'immagine *blurred* X.

Dopo averla filtrata con la finestra di Hann, viene effettuata la stima della PSF come descritto nella sezione (3.2.1).

Infine, dopo aver stimato i parametri del *kernel*, viene chiamata la routine di *deblurring non-blind* descritta nella sezione (3.1).

Di seguito viene presentato lo pseudocodice per effettuare il *deblurring blind*.

Data: Immagine *blurred* y, fattore di regolarizzazione λ

Result: Immagine *deblurred* X

1 Determinare i parametri della PSF ($LENGTH$,θ) di y;

2 Rendere y vettore colonna;

3 Creare la matrice di *blur* PSF tramite i parametri stimati;

4 Creare la matrice di filtraggio H utilizzando PSF;

5 Creare la matrice di ricostruzione *Wavelet* W_r;

6 $A = H * W_r$;

7 Risoluzione di $\|y - Ax\|_2^2 + \lambda\|x\|_1$ utilizzando la routine GPSR o L1_Ls;

8 $X = reshape(W_r x)$

Algorithm 2: Deblurring blind

Capitolo 4

Risultati Sperimentali

In questo capitolo, vengono presentati i risultati ottenuti applicando gli algoritmi di *deblurring* definiti nel capitolo precedente, paragonandoli a quelli "stato dell'arte" già implementati in MATLAB, per valutarne l'efficienza.

Oltre a una valutazione qualitativa, ne viene data una quantitativa tramite il valore *Peak Signal to Noise Ratio* (PSNR). Sia X l'immagine originale e Y quella ricoverata, entrambe di dimensioni $M \times N$, si calcola il PSNR, in decibel, definendo l'*Errore Quadratico Medio* (MSE) come:

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \|X(i,j)Y(i,j)\|^2 \tag{4.1}$$

Il PSNR è definito da:

$$PSNR(X,Y) = 20 \log_{10} \left(\frac{\max\{X\}}{\sqrt{MSE}} \right) \tag{4.2}$$

4.1 Deblurring non-blind

I test nel caso di *deblurring non-blind* sono stati eseguiti su immagini di dimensioni 256×256 pixel alle quali è stato applicato il *motion blur* lineare, tramite la funzione *fspecial* di MATLAB, di lunghezza 20 o 25 pixel e l'angolo

che varia tra $0° - 180°$. È stato inoltre aggiunto del rumore additivo *gaussiano* con varianza 0.002 e media nulla.

Per risolvere il problema della minimizzazione, sono stati utilizzati i pacchetti GPSR[9] e L1_Ls[10], utilizzando rispettivamente i metodi *GPSR_BB* e *l1_ls*. Durante l'applicazione di tali metodi, sono stati variati i parameri *lambda* e *tollerance* assegnando dei valori, rispettivamente, tra $\lambda = [0.1 - 0.001]$ e $tol = [10 - 0.0001]$.

Per quanto riguarda i metodi classici di deconvoluzione, sono stati utilizzati quelli già implementati in MATLAB, cioè:

- *deconvlucy* che esegue il metodo *Lucy-Richardson.*[14]

  In questo caso, i parametri che sono passati all'algoritmo sono, oltre all'immagine degradata e la PSF, il numero di iterazioni che vengono eseguite dall'algoritmo.

- *deconvwnr* che esegue il metodo del *filtro di Wiener.*[15] Alla routine, viene passato in ingresso il rapporto segnale-rumore, chiamato *Noise to Signal Ratio*, cioè:

  $$SNR = \frac{P_{segnale}}{P_{rumore}}$$

  dove $P_{segnale}$ è la potenza del segnale, mentre P_{rumore} è quella del rumore.

- *deconvblind* che esegue la deconvoluzione *blind*[16], al quale, come stima iniziale della PSF, viene passata in ingresso la PSF reale. Inoltre, a tale metodo, viene assegnato il numero di iterazioni da eseguire durante il suo processo.

Le immagini utilizzate per effettuare i test sono *Lena*, *Text* e *Scimmia.*

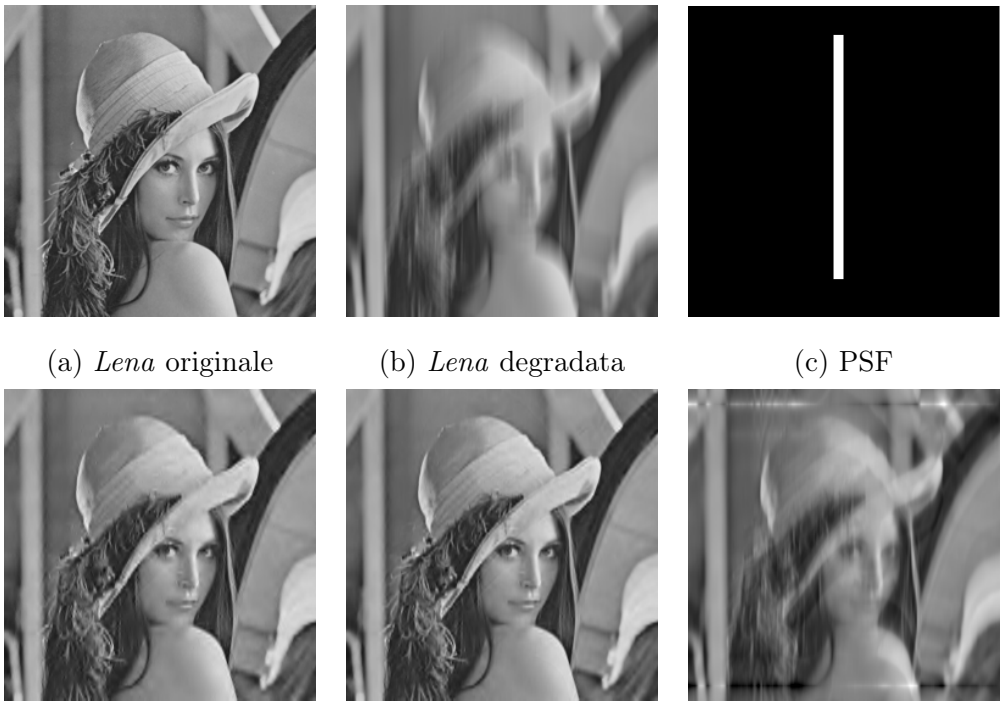
Di seguito, per ciascuna immagine, viene mostrato il risultato della ricostruzione utilizzando il metodo implementato e i metodi classici citati sopra.

(a) *Text* originale (b) *Text* degradata (c) PSF

(d) GPSR (e) L1_Ls (f) *deconvwnr*

(g) *deconvlucy* (h) *deconvblind*

Figura 4.1: Immagine *Text* degradata con $L = 20$ e $\theta = 130°$ e le ricostruzioni con il metodo implementato (d)(e) e i metodi classici (f)(g)(h)

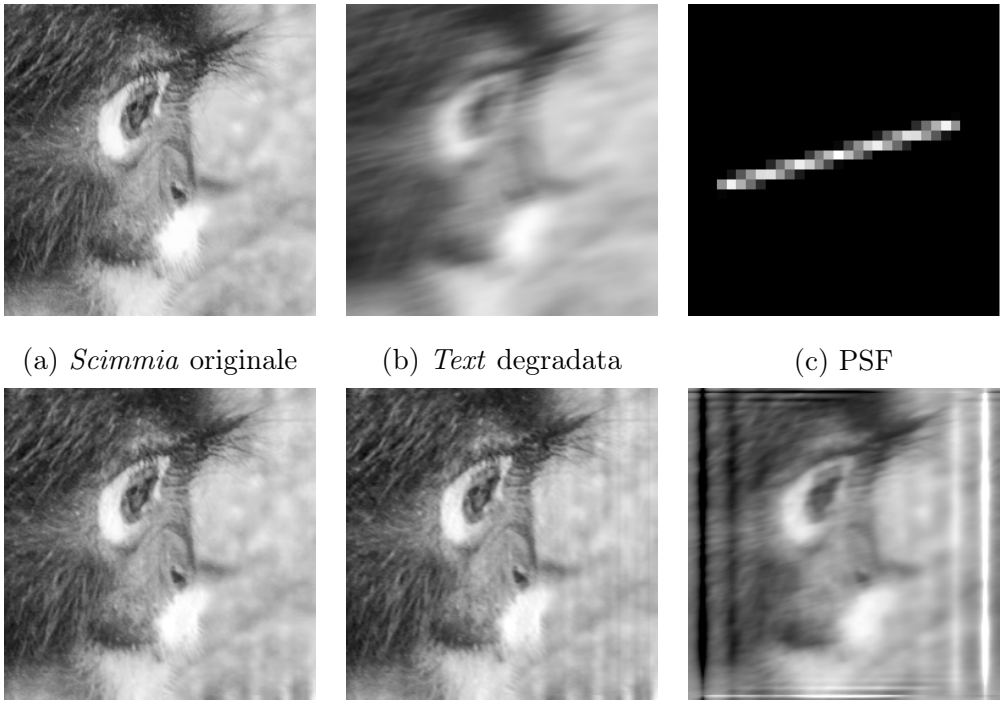


(a) *Lena* originale (b) *Lena* degradata (c) PSF

(d) GPSR (e) L1_Ls (f) *deconvwnr*

(g) *deconvlucy* (h) *deconvblind*

Figura 4.2: Immagine *Lena* degradata con $L = 25$ e $\theta = 90°$ e le ricostruzioni con il metodo implementato (d)(e) e i metodi classici (f)(g)(h)



(a) *Scimmia* originale (b) *Text* degradata (c) PSF

(d) GPSR (e) L1_Ls (f) *deconvwnr*

(g) *deconvlucy* (h) *deconvblind*

Figura 4.3: Immagine *Scimmia* degradata con $L = 25$ e $\theta = 15°$ e le ricostruzioni con il metodo implementato e i metodi classici

	GPSR		**L1_Ls**	
Image	PSNR	Time	PSNR	Time
Text	24.11	315	26.70	310
Lena	31.07	203	33.36	169
Scimmia	33.81	121	30.61	113

Tabella 4.1: PSNR e Tempo di esecuzione (in secondi) del metodo implementato delle le immagini mostrate sopra

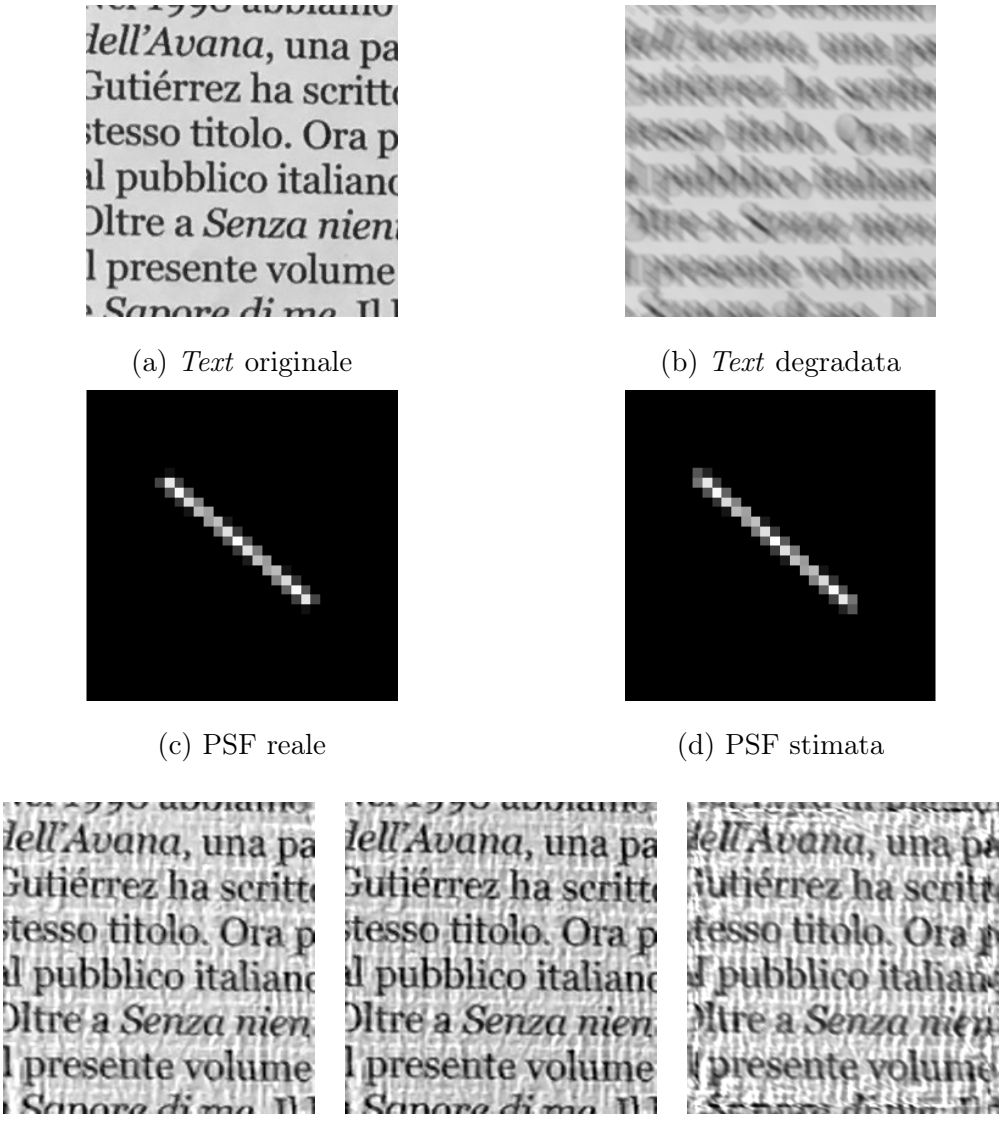
	Wiener Filter		**Lucy-Richardson**		**Blind Deconvolution**	
Image	PSNR	Time	PSNR	Time	PSNR	Time
Text	16.88	0.05	17.74	0.31	17.65	0.84
Lena	23.30	0.15	20.86	0.72	20.68	1.13
Scimmia	20.56	0.14	16.14	0.87	16.03	0.56

Tabella 4.2: PSNR e Tempo di esecuzione (in secondi) dei metodi classici delle le immagini mostrate sopra

4.2 Deblurring blind

I test nel caso di *deblurring blind* sono stati fatti su immagini 256×256 pixel alle quali, come nel caso precedente, è stato artificialmente inserito il *motion blur*. A differenza del metodo precedente, la PSF viene stimata e con quest'ultima viene eseguita la deconvoluzione.

Per ultimo, viene effettuato un test su un'immagine reale di dimensioni 128×128 pixel alla quale è applicato il metodo implementato di *deblurring non blind*.



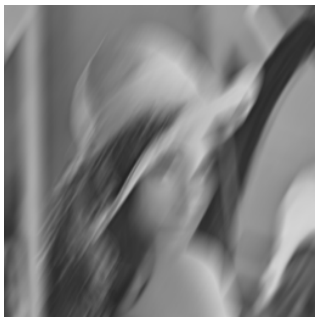
(a) *Text* originale (b) *Text* degradata

(c) PSF reale (d) PSF stimata

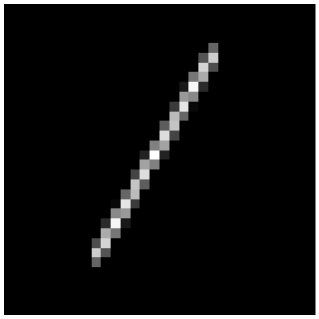
(e) GPSR (f) L1_Ls (g) *deconvblind*

Figura 4.4: Immagine *Text* degradata con $L = 20$ e $\theta = 140°$ (c), la PSF stimata con $L = 21$ e $\theta = 140°$ (d) e le ricostruzioni con il metodo implementato (e)(f) e con quello di restauro *blind* di MATLAB (g)

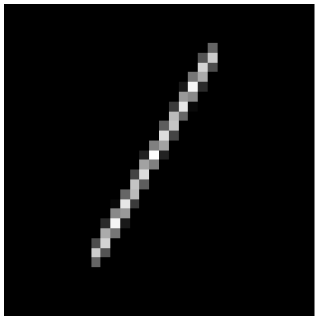
(a) *Lena* originale



(b) *Lena* degradata



(c) PSF reale



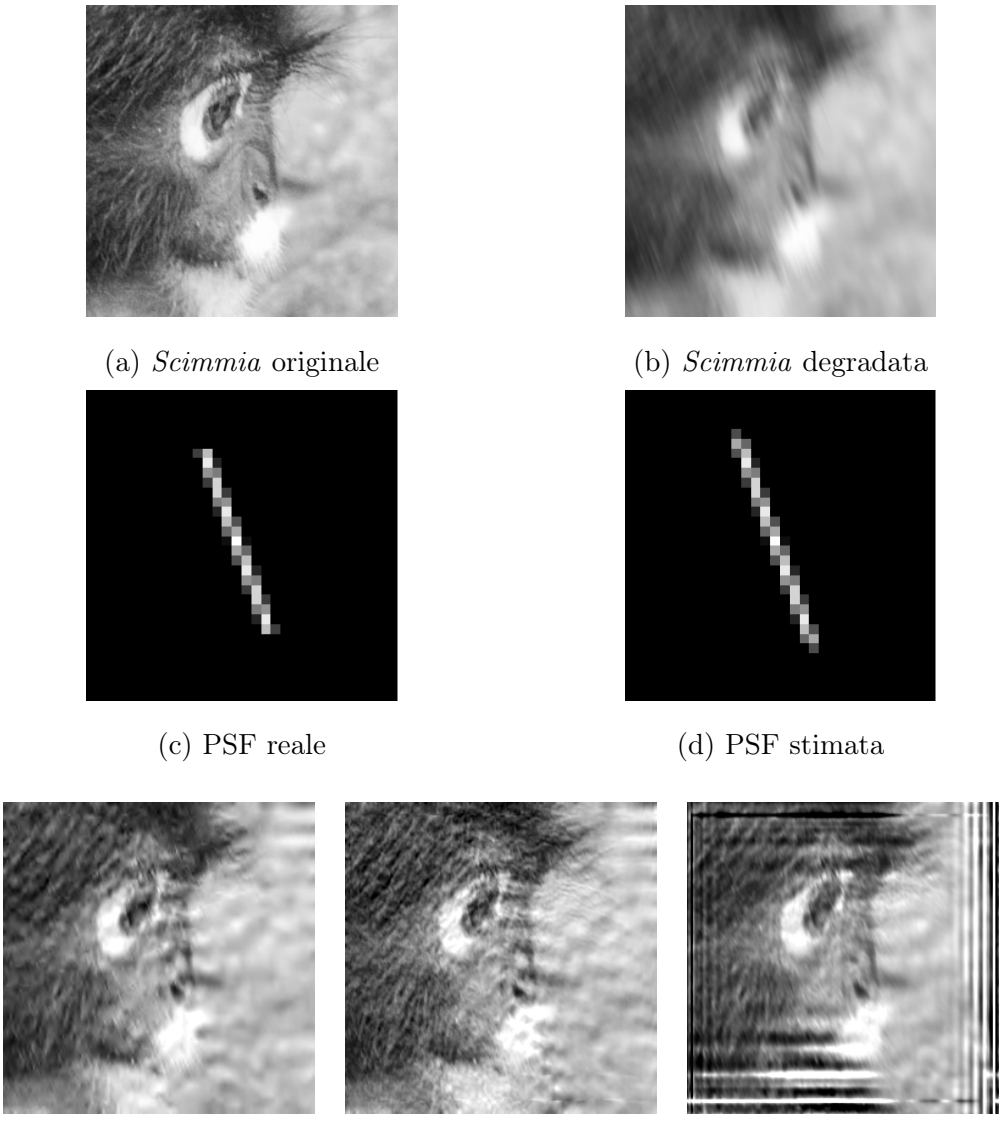
(d) PSF stimata

(e) GPSR

(f) L1_Ls

(g) *deconvblind*

Figura 4.5: Immagine *Lena* degradata con $L = 25$ e $\theta = 60°$ (c), la PSF stimata con $L = 20$ e $\theta = 60°$ (d) e le ricostruzioni con il metodo implementato (e)(f) e con quello di restauro *blind* di MATLAB (g)



(a) *Scimmia* originale (b) *Scimmia* degradata

(c) PSF reale (d) PSF stimata

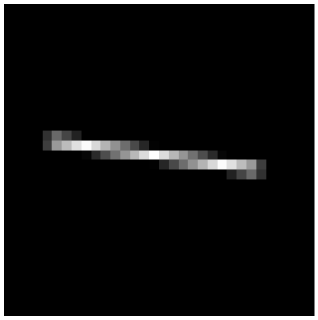
(e) GPSR (f) L1_Ls (g) *deconvblind*

Figura 4.6: Immagine *Scimmia* degradata con $L = 20$ e $\theta = 110°$ (c), la PSF stimata con $L = 23$ e $\theta = 110°$ (d) e le ricostruzioni con il metodo implementato (e)(f) e con quello di restauro *blind* di MATLAB (g)

	GPSR		**L1_Ls**		**Blind Deconvolution**	
Image	PSNR	Time	PSNR	Time	PSNR	Time
Text	19.95	238	19.24	302	16.71	0.52
Lena	29.18	313	30.69	58	19.24	0.62
Scimmia	24.41	124	23.54	186	14.79	0.58

Tabella 4.3: PSNR e Tempo di esecuzione (in secondi) del metodo implementato e del metodo di deconvoluzione *blind* di MATLAB delle le immagini mostrate sopra

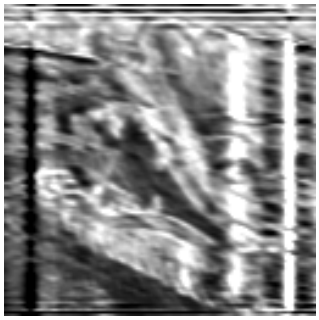
(a) *Thai* degradata



(b) PSF stimata

(c) GPSR

(d) L1_Ls



(e) Blind deblurring

Figura 4.7: Immagine *Thai* degradata, la PSF stimata con $L = 22$ e $\theta = 172°$ (b) e le ricostruzioni con il metodo implementato (c)(d) e con quello di restauro *blind* di MATLAB (e)

Appendice A

Trasformata Wavelet

La *trasformata Wavelet* è una trasformata utilizzata per l'analisi, la compressione e la trasmissione delle immagini. Questa si basa su piccole onde, chiamate *Wavelet*, di frequenza variabile e durata limitata.

A.1 Funzioni di scaling

Si consideri l'insieme delle funzioni di espansione composto dalle traslazioni intere e dalle riduzioni in scala binarie della funzione reale integrabile al quadrato $\phi(x)$; questo è l'insieme $\{\phi_{j,k}(x)\}$, dove

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k) \tag{A.1}$$

per tutti i valori $j, k \in \mathcal{Z}$ e $\phi(x) \in L^2(\mathcal{R})$[1]. In questo caso k determina la posizione di $\phi_{j,k}(x)$ lungo l'asse delle x e j ne controlla l'ampiezza, determinata dal termine $2^{j/2}$. Dato che la funzione $\phi(x)$ cambia con j, questa viene definita *funzione di scaling*.

[1]La notazione $L^2(\mathcal{R})$ indica l'insieme delle funzioni reali unidimensionali, misurabili e integrabili al quadrato.

Per un determinato $j = j_0$ si indica il sottospazio[2] come

$$V_{j_0} = \overline{Span_k \phi_{j_0,k}(x)} \tag{A.2}$$

Se $f(x) \in V_{j_0}$, possiamo scrivere

$$f(x) = \sum_k \alpha_k \phi_{j_0,k}(x) \tag{A.3}$$

A.2 Funzioni Wavelet

Data una *funzione di scaling* possiamo definire una *funzione Wavelet* $\psi(x)$ che, insieme alle sue traslazioni intere e alle sue riduzioni in scala binarie, ricopre la differenza tra due sottospazi di *scaling* adiacenti V_j e V_{j+1}.

Definiamo l'insieme $\{\psi_{j,k}(x)\}$ delle Wavelet

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \tag{A.4}$$

per tutti i valori $k \in \mathcal{Z}$ che coprono gli spazi W_j. Come con le *funzioni di scaling*, scriviamo

$$W_j = \overline{Span_k \{\psi_{j,k}(x)} \tag{A.5}$$

e notiamo che se $f(x) \in W_j$,

$$f(x) = \sum_k \alpha_k \psi_{j,k}(x) \tag{A.6}$$

A.3 Trasformate Wavelet in una dimensione

Come nel *dominio di Fourier*, è possibile definire alcune trasformate Wavelet strettamente collegate: l'*espansione in serie Wavelet*, la *trasformata Wavelet discreta* e la *trasformata Wavelet continua*. In relazione al presente lavoro, verrà approfondita solo la ltrasformata discreta mentre per approfondimenti si può fare riferimento a [1].

[2]In generale, denoteremo il sottospazio su k per ogni valore j come $V_j = \overline{Span_k \phi_{j,k}(x)}$

A.3.1 Trasformata Wavelet discreta

Come quella di *Fourier*, l'*espansione in serie Wavelet* fa corrispondere una funzione di variabile continua a una sequenza di coefficienti. Se la funzione da espandere è *discreta*, i coefficienti che ne risultano determinano la sua *trasformata Wavelet discreta* (*DWT*).

Considerando la funzione *discreta*

$$f(n) = f(x_0 + n\Delta x)$$

per alcuni valori x_0, Δx ed $n = 0, 1, \ldots, M-1$, i coefficienti di espansione in *serie Wavelet* per $f(x)$ diventano i coefficienti DWT *diretti* per la sequenza $f(n)$:

$$W_\phi(j_0, k) = \frac{1}{\sqrt{M}} \sum_n f(n)\phi_{j_0,k}(n) \tag{A.7}$$

$$W_\psi(j, k) = \frac{1}{\sqrt{M}} \sum_n f(n)\psi_{j,k}(n) \quad \text{per } j \geq j_0 \tag{A.8}$$

La DWT *inversa* è definita da

$$f(n) = \frac{1}{\sqrt{M}} \sum_k W_\phi(j_0, k)\phi_{j_0,k}(n) + \sum_{j=j_0}^{\infty} \sum_k W_\psi(j, k)\psi_{j,k}(n) \tag{A.9}$$

A.4 Trasformata Wavelet in due dimensioni

Per definire la trasformata bidimensionale, sono necessarie una *funzione di scaling bidimensionale* $\phi(x, y)$ e tre *Wavelet bidimensionali* $\psi^H(x, y)$, $\psi^V(x, y)$ e $\psi^D(x, y)$ dove ognuna è un prodotto di due funzioni bidimensionali.

Escludendo i prodotti che determinano risultati monodimensionali (come $\phi(x)\psi(x)$), i quattro rimanenti producono la *funzione di scaling separabile*

$$\phi(x, y) = \phi(x)\phi(y) \tag{A.10}$$

e le *Wavelet separabili*

$$\psi^H = \psi(x)\phi(y) \tag{A.11}$$

$$\psi^V = \phi(x)\psi(y) \tag{A.12}$$

$$\psi^D = \psi(x)\psi(y) \tag{A.13}$$

Queste *Wavelet*, per le immagini, misurano variazioni di intensità lungo direzioni diverse: ψ^H misura le variazioni lungo le colonne (ad esempio picchi orizzontali), ψ^V risponde alle variazioni lungo le righe (come picchi verticali) e ψ^D corrisponde a variazioni diagonali.

Si possono derivare le *funzioni di scaling* e *Wavelet bidimensionali* e *separabili* partendo dalla DWT 1-D. Si definiscono le funzioni di base scalate e traslate:

$$\phi_{j,m,n}(x,y) = 2^{j/2}\phi(2^j x - m, 2^j y - n) \tag{A.14}$$

$$\psi^i_{j,m,n}(x,y) = 2^{j/2}\psi^i(2^j x - m, 2^j y - n) \quad \text{con } i = \{H, V, D\} \tag{A.15}$$

La *trasformata Wavelet discreta* dell'immagine $f(x,y)$ di dimensioni $M \times N$ è quindi

$$W_\phi(j_0,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\phi_{j_0,m,n}(x,y) \tag{A.16}$$

$$W^i_\psi(j,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\psi^i_{j,m,n}(x,y) \tag{A.17}$$

Come nel caso monodimensionale, j_0 è una scala di partenza arbitraria e i coefficienti $W_\phi(j_0,m,n)$ definiscono un'approssimazione di $f(x,y)$ alla scala j_0. I coefficienti $W^i_\psi(j_0,m,n)$ aggiungono dei dettagli orizzontali, verticali e diagonali per le scale $j \geq j_0$. Normalmente poniamo $j_0 = 0$ e selezioniamo $N = M = 2^j$ cosicché $j = 0, 1, 2, \ldots, J-1$ e $m = n = 0, 1, 2, \ldots, 2^j - 1$.

Dati W_ϕ e W_ψ^i delle equazioni (A.16) e (A.17), $f(x,y)$ è ottenuta tramite la *trasformata Wavelet discreta*

$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_m \sum_n W_\phi(j_0,m,n)\phi_{j_0,m,n}(x,y) + \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_0}^{\infty} \sum_m \sum_n W_\psi^i(j,m,n)\psi_{j,m,n}^i(x,y)$$

Bibliografia

[1] Rafael C. Gonzales; Richard E. Woods. *Elaborazione delle immagini digitali*. A cura di Pearson. 2008. P. 809. (Cit. alle pp. 3, 8, 37, 52).

[2] James G. Nagy Per Christian Hansen e Dianne P. O,ÄôLeary. *Deblurring Images: Matrices, Spectra, and Filtering*. A cura di SIAM. 2006. P. 130. (Cit. a p. 7).

[3] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964. (Cit. a p. 9).

[4] L. B. Lucy. "An iterative technique for the rectification of observed distributions". *The Astronomical Journal* 79, pp. 745–754, 1974. (Cit. a p. 12).

[5] William H. Richardson. "Bayesian-Based Iterative Method of Image Restoration". *Journal of the Optical Society of America* 62, pp. 55–59, 1972. (Cit. a p. 12).

[6] Z. Hu, J. B. Huang e M. H. Yang. "Single image deblurring with adaptive dictionary learning". Pp. 1169–1172, 2010. DOI: 10.1109/ICIP.2010.5651892. (Cit. a p. 17).

[7] Weisheng Dong, Lei Zhang, Guangming Shi e Xiaolin Wu. "Image Deblurring and Super-Resolution by Adaptive Sparse Domain Selection

and Adaptive Regularization". *Trans. Img. Proc.* 20, pp. 1838–1857, 2011. (Cit. a p. 17).

[8] J. F. Cai, H. Ji, C. Liu e Z. Shen. "Framelet-Based Blind Motion Deblurring From a Single Image". *IEEE Transactions on Image Processing* 21, pp. 562–572, 2012. DOI: 10.1109/TIP.2011.2164413. (Cit. a p. 19).

[9] M. A. T. Figueiredo, R. D. Nowak e S. J. Wright. "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems". *IEEE Journal of Selected Topics in Signal Processing* 1, pp. 586–597, 2007. DOI: 10.1109/JSTSP.2007.910281. (Cit. alle pp. 25, 32, 42).

[10] Kwangmoo Koh, Seungjean Kim e Stephen Boyd. "l1 ls: A Matlab Solver for Large-Scale l1-Regularized Least Squares Problems". 2007. (Cit. alle pp. 25, 33, 42).

[11] MATLAB. *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010. (Cit. alle pp. 25, 33).

[12] João Oliveira, Mário Figueiredo e José Bioucas-Dias. "Blind estimation of motion blur parameters for image deconvolution". *Pattern Recognition and Image Analysis*, pp. 604–611, 2007. (Cit. a p. 39).

[13] Shamik Tiwari, Vidya Prasad Shukla e Ajay Singh Sangappa Biradar. "Review of Motion Blur Estimation Techniques". *Journal of Image and Graphics* 1, pp. 176–184, 2013. (Cit. a p. 39).

[14] MATLAB. *deconvlucy*. 2006. URL: http://it.mathworks.com/help/images/ref/deconvlucy.html (cit. a p. 42).

[15] MATLAB. *deconvwnr*. 2006. URL: http://it.mathworks.com/help/images/ref/deconvwnr.html (cit. a p. 42).

[16] MATLAB. *deconvblind*. 2006. URL: `http://it.mathworks.com/help/images/ref/deconvblind.html` (cit. a p. 42).