RICOSTRUZIONE DI UN'IMMAGINE SFOCATA UTILIZZANDO I FILTRI DI GABOR E L'ALGORITMO *INDEPENDENT COMPONENT ANALYSIS*

GIACOMO VITALI E MARCO LA GALA

Progetto didattico per il corso di Analisi Numerica a.a. 2017/2018 - Prof. Simone Fiori

Sommario. Lo scopo di questa relazione è presentare un metodo che permette di ricostruire un'immagine originale partendo da un'immagine sfocata e di confrontare le differenze tra l'algoritmo *Independent Component Analysis* (ICA) presentato da Shinji Umeyama [1] e una sua variante che utilizza le proprietà dei Manifold su una sfera unitaria.

Key words. Ricostruzione di un'immagine sfocata, Filtri di Gabor, Algoritmo Independent Component Analysis, Proprietà dei Manifold.

1. Introduzione. Il processo di ricostruzione di un'immagine consiste nel trovare l'immagine originale partendo da una sua versione sfocata e con disturbi. La sfocatura viene solitamente modellata tramite la convoluzione di un *Point Spread Function* (PSF) con l'immagine originale aggiungendo un ulteriore disturbo. Come è possibile notare nella Figura 1.1 l'immagine originale f dopo essere stata modificata dal PSF diventa un'immagine g sfocata.

Se il PSF è conosciuto a priori si trovano diversi algoritmi che permettono di ricostruire l'immagine mentre, se non è già noto, la maggior parte degli algoritmi non funziona correttamente in quanto non è semplice stimare contemporaneamente all'immagine anche il PSF. Esiste un tipo di algoritmo chiamato *Blind Deconvolution* il quale consente di ripristinare l'immagine originale stimando contemporaneamente sia l'immagine che il PSF. Viene principalmente utilizzato per la ricostruzione delle immagini in astronomia dove si ha la necessità, infatti, di migliorare le fotografie ottenute dai telescopi terrestri le quali risultano degradate dalle turbolenze atmosferiche.

Umeyama, nel suo metodo proposto e discusso in [1] e [2], ha applicato i filtri di Gabor all'immagine sfocata che si deve ricostruire g, in modo da scomporla in un numero di immagini filtrate (*Gabor filter 1, . . . , Gabor filter M*). Quest'ultime, insieme all'immagine sfocata g, vengono utilizzate in input dall'algoritmo ICA che permette di ottenere un'immagine f' simile all'originale eliminando la sfocatura.

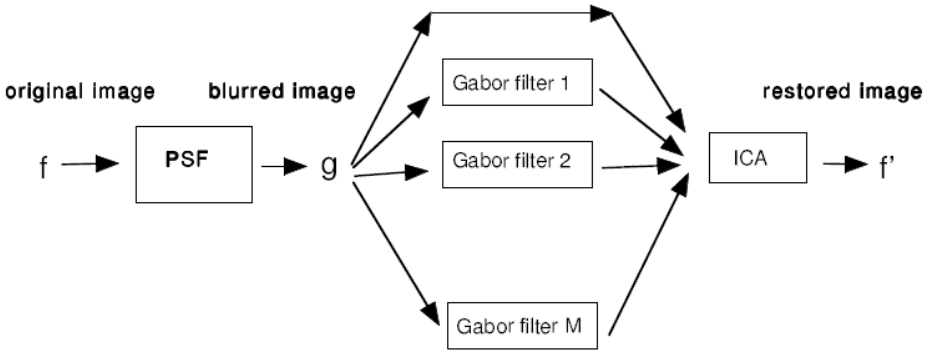

FIGURA 1.1. *Schema dei passi del metodo proposto da Umeyama [1].*

Negli ultimi anni questo tipo di algoritmo è stato ampiamente studiato soprattutto nel campo

dell'analisi dei dati unidimensionali come per esempio il suono. Nel documento di Umeyama, invece, viene fornita una nuova applicazione per l'analisi dei dati bidimensionali la quale permette di ricostruire un'immagine partendo da una sua versione sfocata senza conoscere il PSF utilizzato. Oltre ad implementare la versione originale dell'algortimo, abbiamo modificato l'algoritmo ICA presente nel documento di Umeyama [1] con le nostre conoscenze riguardo i Manifold utilizzando i concetti relativi alla sfera unitaria e alla mappa esponenziale.

Di seguito viene fornita una spiegazione dei due metodi utilizzati corredata da esempi con alcune immagini di prova ed un esperimento utilizzando un'immagine realmente sfocata.

2. Teoria e metodi.

2.1. Filtri di Gabor. I filtri di Gabor, ampiamente utilizzati nel campo della computer vision, realizzano filtri multicanale che possono scomporre un'immagine in un numero di immagini filtrate. I filtri di Gabor di due dimensioni sono costruiti come il prodotto di una funzione Gaussiana 2D e un'onda piana che si propaga su un piano 2D. Ogni filtro è determinato dalla deviazione standard della funzione gaussiana dalla direzione di propagazione e dalla lunghezza d'onda di un'onda piana.

I filtri di Gabor in 2D sono definiti come una funzione complessa, dove le parti reali e immaginarie vengono considerate come due filtri reali distinti:

$$R(x, y; \nu, k) = \exp\left(-\frac{x^2 + y^2}{2{\sigma_\nu}^2}\right) \cdot \cos\frac{\pi}{\sigma_\nu}\left(x\cos\phi_k + y\sin\phi_k\right), \tag{2.1}$$

$$I(x, y; \nu, k) = \exp\left(-\frac{x^2 + y^2}{2{\sigma_\nu}^2}\right) \cdot \sin\frac{\pi}{\sigma_\nu}\left(x\cos\phi_k + y\sin\phi_k\right), \tag{2.2}$$

dove

$$\sigma_\nu = \left(\sqrt{2}\right)^{\nu+1} \qquad \text{e} \qquad \phi_k = \frac{\pi}{4}k. \tag{2.3}$$

Per i nostri esempi abbiamo utilizzato $\nu \in \{\, 0, 1 \,\}$ e $k \in \{\, 0, 1, 2, 3 \,\}$ come parametri. Otteniamo cos 16 filtri di Gabor: 8 reali e 8 immaginari. La Figura 2.1 mostra i filtri che abbiamo implementato stampati come immagini. Le prime due righe mostrano quelli ottenuti imponendo $\nu = 0$ mentre le altre due i filtri con $\nu = 1$. La prima e la terza riga, inoltre, mostrano i filtri reali $R(x, y; \nu, k)$ che simulano la risposta di un semplice neurone con la capacit di riconoscere una linea mentre la seconda e la quarta quelli immaginari $I(x, y; \nu, k)$ che a loro volta simulano la risposta di un semplice neurone con la capacit di riconoscere un bordo.

Nelle espressioni (2.1) e (2.2), le coppie (x, y) rappresentano le coordinate dei pixel di un'immagine nella forma (colonna, riga).

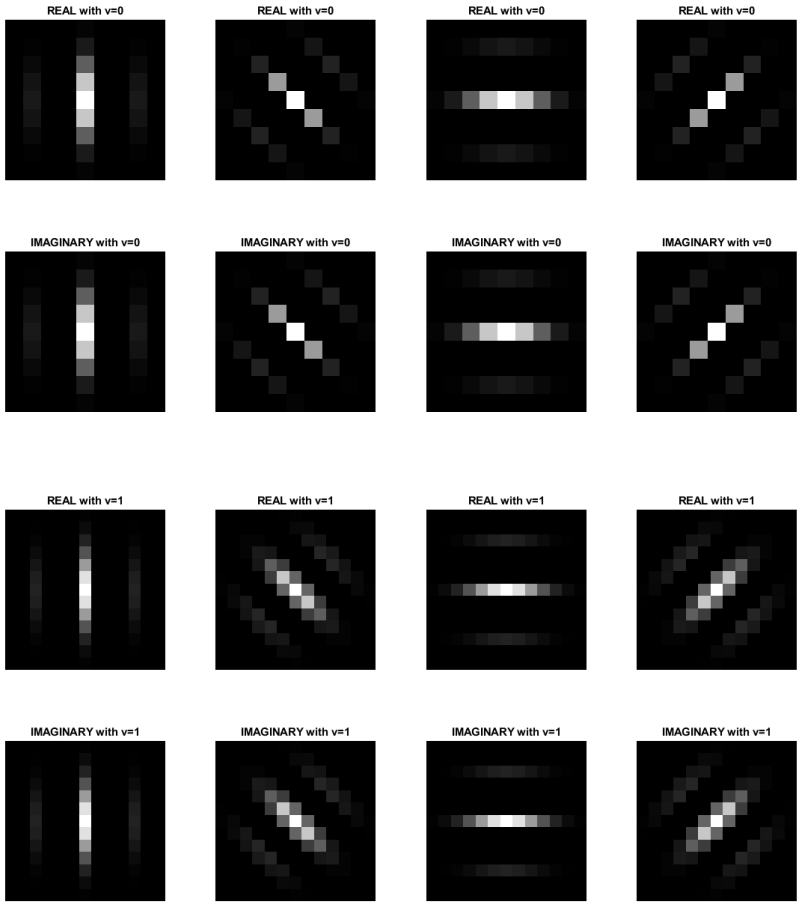

FIGURA 2.1. *Filtri di Gabor definiti nella Sezione 2.1 con* $M = 6$.

2.2. ***Blind deconvolution*** **di un'immagine sfocata usando i filtri di Gabor.** Il processo per sfocare un'immagine consiste in una convoluzione tra l'immagine originale $f(x, y)$ e un PSF $h(x, y)$. Non andremo a considerare il caso in cui l'immagine sfocata da ricostruire abbia, oltre alla sfocatura, un disturbo non prevedibile come per esempio il pulviscolo atmosferico o le polveri sottili presenti nell'aria al momento della foto.

Si può definire il valore di un pixel $g(x, y)$ dell'immagine sfocata come la somma dei valori dei pixel adiacenti attraverso la convoluzione discreta:

$$g(x, y) = \sum_{s=-M}^{M} \sum_{t=-M}^{M} h(s, t) f(x + s, y + t) \tag{2.4}$$

dove M rappresenta l'estensione del PSF ($2M + 1$ righe e colonne). Un esempio mostrato nella Figura 2.2. Allo stesso tempo $f(x + s, y + t)$, che rappresenta il valore di un pixel adiacente, può essere rappresentato come la somma dei valori del pixel centrale $f(x, y)$ e delle sue derivate $f_x(x, y)$

FIGURA 2.2. *Immagini ottenute applicando i filtri di Gabor ad un'immagine sfocata.*

e $f_y(x, y)$ tramite l'espansione di Taylor.

$$f(x + s, y + t) = f(x, y) + \alpha(sf_x(x, y) + tf_y(x, y) + \dots). \tag{2.5}$$

Usando le equazioni (2.4) e (2.5) il valore del pixel $g(x, y)$ può essere approssimato come segue:

$$g(x, y) = a_1 f(x, y) + a_2 f_x(x, y) + a_3 f_y(x, y) + \dots \tag{2.6}$$

dove si sono utilizzati i coefficenti

$$a_1 = \sum_{s=-M}^{M} \sum_{t=-M}^{M} h(s,t), \tag{2.7}$$

$$a_2 = \alpha \sum_{s=-M}^{M} \sum_{t=-M}^{M} s \cdot h(s,t), \tag{2.8}$$

$$a_3 = \alpha \sum_{s=-M}^{M} \sum_{t=-M}^{M} t \cdot h(s,t), \tag{2.9}$$

$$\dots$$

Nel nostro caso specifico abbiamo applicato i filtri di Gabor all'immagine sfocata. Significa che all'immagine originale viene applicata la convoluzione due volte: una con i PSF e l'altra con i filtri di Gabor. Ma questo equivale a fare la convoluzione tra l'immagine originale e una funzione $h' = R * h$ o $h' = I * h$ dove R e I sono i filtri di Gabor rispettivamente reali e immaginari.

Quindi possiamo riscrivere, sulle basi delle considerazioni precedenti, l'equazione dell'immagine filtrata con i filtri di Gabor $g'(x,y)$.

$$g'(x,y) = a'_1 f(x,y) + a'_2 f_x(x,y) + a'_3 f_y(x,y) + \dots \tag{2.10}$$

dove si sono utilizzati i nuovi coefficenti

$$a'_1 = \sum_{s=-M}^{M} \sum_{t=-M}^{M} h'(s,t), \tag{2.11}$$

$$a'_2 = \alpha \sum_{s=-M}^{M} \sum_{t=-M}^{M} s \cdot h'(s,t), \tag{2.12}$$

$$a'_3 = \alpha \sum_{s=-M}^{M} \sum_{t=-M}^{M} t \cdot h'(s,t), \tag{2.13}$$

$$\dots$$

I coefficenti a'_i dipendono dalla forma del filtro di Gabor che abbiamo applicato.

Le equazioni (2.5) e (2.10) ci mostrano che l'immagine g' contiene l'immagine originale f e le sue derivate f_x, f_y. Questo ci porta a pensare che sia possibile ripristinare l'immagine originale (e le sue derivate) quando applichiamo l'ICA alle immagini doppiamente sfocate (sia col PSF che con i filtri di Gabor).

2.3. Algoritmo ICA. Trasformiamo le immagini ottenute (le 16 con i filtri di Gabor e quella sfocata) che non sono altro che matrici $n \times n$, in vettori g_i di dimensione $n^2 \times 1$ tramite una scansione da sinistra a destra e dall'alto verso il basso ottenendo così 17 vettori g_i^T, con $i = 1, \dots, 17$, i quali andranno a formare una matrice X di dimensione $17 \times n^2$ della seguente forma:

$$X = \begin{bmatrix} {g_1}^T \\ {g_2}^T \\ \dots \\ {g_{17}}^T \end{bmatrix} \tag{2.14}$$

Da notare che le righe che compongono la matrice X sono le 16 immagini ottenute dai filtri e l'immagine sfocata, mentre il numero delle colonne n^2 rappresenta il numero totale di pixel di ogni immagine.

Prima di applicare l'algoritmo ICA alla matrice delle immagini X è utile modificarla attraverso due operazioni chiamate *centering* e *whitening* in modo tale da ottenere migliori prestazioni dell'algoritmo.

L'operazione di *centering* consiste nel rendere le colonne della matrice X a valor medio nullo. Scomponiamo, quindi, la matrice X nelle sue n^2 colonne, ovvero:

$$X = [x_1\ x_2\ x_3\ x_4\ \cdots\ x_{n^2}], \tag{2.15}$$

dove ciascun vettore x_k ha dimensione 17×1. Il valore medio delle n^2 colonne della matrice X si calcola come:

$$m := \frac{1}{n^2} \sum_{k=1}^{n^2} x_k, \tag{2.16}$$

e le colonne centrate si definiscono come

$$\check{x}_k := x_k - m. \tag{2.17}$$

Indichiamo con $\check{X}$ la matrice le cui colonne sono date dai vettori $\check{x}_k$.

L'operazione di *whitening* consiste nel trasformare linearmente ciascuna colonna della matrice $\check{X}$ in modo da ottenere una nuova matrice $Z = [z_1\ z_2\ z_3\ \cdots z_{n^2}]$ le cui colonne abbiano covarianza unitaria. Si noti che la matrice Z ha dimensione $17 \times n^2$. La matrice di covarianza associata alle colonne della matrice Z si definisce come

$$C_z := \frac{1}{n^2} \sum_{k=1}^{n^2} z_k z_k^T. \tag{2.18}$$

Si noti che la matrice C_z ha dimensione 17×17. Si può verificare che la stessa matrice di covarianza si può calcolare come

$$\frac{1}{n^2} Z Z^T = \frac{1}{n^2} (z_1 z_1^T + z_2 z_2^T + \cdots + z_{n^2} z_{n^2}^T). \tag{2.19}$$

Uno dei metodi diffusi per il whitening è quello di usare la decomposizione degli autovalori (EVD) della matrice di covarianza $C_z = EDE^T$, dove E è la matrice ortogonale degli autovettori di C_z e D è la matrice diagonale degli autovalori $D = diag(d_1, \ldots, d_n)$.

Il whitening allora può essere calcolato con la seguente formula

$$Z = D^{-1/2} E^T \check{X} \tag{2.20}$$

dove la matrice $D^{-1/2}$ è calcolata come $D^{-1/2} = diag\left(d_1^{-1/2}, \ldots, d_n^{-1/2}\right)$.

Prima di applicare la Formula (2.20) abbiamo avuto l'accortezza di eliminare i possibili autovalori inferiori a 0.0001 della matrice X, ottenuta dopo l'operazione di *centering*, applicando un restringimento. In particolare abbiamo eliminato i suoi possibili autovalori negativi dovuti a errori numerici. Otteniamo così una matrice Z di dimensione $l \times n^2$ dove n^2 rappresenta il numero di pixel dell'immagine utilizzata ed l il numero di autovalori presi con il restringimento.

L'algoritmo *Independent Component Analysis* utilizza in input la matrice Z che contiene le varie immagini ottenute dai filtri di Gabor e l'immagine sfocata con le opportune variazioni dovute al *centering* e al *whitening*. La Formula (2.21), la quale è la relazione ingresso-uscita di una rete neurale con pesi w, ci permette di ottenere l'immagine restaurata trovando la prima componente indipendente P e consiste nell'applicare il prodotto tra una matrice w^T (di dimensione $1 \times l$) che contiene i pesi di una rete neurale w e una matrice Z ottenuta precedentemente.

$$P = w^T Z \qquad (vettore\ 1 \times n^2) \tag{2.21}$$

dove il vettore w viene modificato con la seguente regola di aggiornamento:

$$w \leftarrow w + \mu Z \tanh(P^T) \tag{2.22}$$

per poi essere normalizzato

$$w \leftarrow \frac{w}{||w||} \tag{2.23}$$

La Formula (2.22) e (2.23) permettono alla rete neurale di apprendere ad eseguire l'algoritmo ICA e vengono iterate più volte fino al raggiungimento di un apprendimento soddisfacente.

Impostare un valore del *learning parameter* μ corretto è molto importante in quanto permette la convergenza veloce e stabile nei metodi di ottimizzazione non lineari. Se w è convergente otteniamo la prima componente indipendente. Non abbiamo bisogno delle altre per il problema della *Blind Deconvolution*.

2.4. Manifold. Il *Manifold*, o *varietà* in italiano, è uno spazio topologico che localmente ha le caratteristiche di uno spazio Euclideo ma che globalmente può avere proprietà geometriche molto differenti da esso, ad esempio la curvatura (la quale non è una proprietà degli spazi Euclidei).

Un esempio di Manifold unidimensionale potrebbe essere il concetto di linea o cerchio, mentre quello più comune di 2-Manifold (ossia di due dimensioni) è la sfera. Infatti la terra, vista (localmente) dai suoi abitanti, potrebbe sembrare un piano, ma dallo spazio (globalmente) acquista la forma di una sfera con proprietà molto diverse da quelle di un piano.

La proprietà più significativa dei Manifold è che sono una densa collezione di dati astratti ai quali possiamo assegnare delle coordinate [5]. Questa proprietà è mostrata in Figura 2.3.

Nella nostra applicazione abbiamo utilizzato un tipo particolare di Manifold chiamato *Ipersfera* (*real hyper-sphere*) [4], il quale è definito in maniera seguente:

$$S^{n-1} = \{x \in \mathbb{R}^n \mid x^T x = 1\} \tag{2.24}$$

Grazie alle conoscenze riguardo i Manifold, abbiamo pensato a un metodo alternativo per l'aggiornamento del vettore dei pesi w in particolare riguardo la Formula (2.22) e (2.23) la quale permette di aggiornare il vettore w per poi normalizzarlo. Visto che il compito dell'algoritmo ICA può essere schematizzato come la ricerca del massimo di una data funzione, lo scopo dell'algoritmo è cercare il punto di massimo di una funzione su un 2-Manifold, dove il vettore dei pesi w attuale viene aggiornato seguendo il gradiente che rappresenta la direzione di massimo incremento della funzione $f(w)$.

Per questo motivo abbiamo utilizzato il concetto di mappa esponenziale:

$$\exp_x(v) = x \cdot \cos||v|| + \frac{v \cdot \sin||v||}{||v||} \tag{2.25}$$

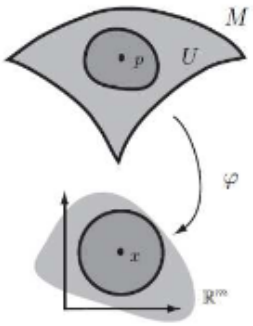

FIGURA 2.3. *Coordinate dei Manifold.*

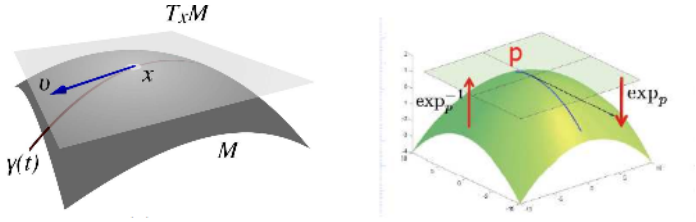

FIGURA 2.4. *Vettore tangente ad un Manifold e mappa esponenziale/mappa logaritmica.*

dove $x \in M$ e il vettore tangente $v \in T_xM$. Applicando la mappa esponenziale si ottiene un nuovo punto $q = \exp_x(v)$ (Figura 2.4).

Utilizziamo anche il concetto di gradiente Riemanniano:

$$\nabla_w f = \left(I - ww^T\right) \frac{\partial f}{\partial w}, \tag{2.26}$$

dove $\frac{\partial f}{\partial w}$ rappresenta la matrice Jacobiana della funzione $f(w)$ da ottimizzare.

Possiamo quindi adattare le Formule (2.25) e (2.26) secondo lo scopo della nostra applicazione, ottenendo i seguenti risultati:

$$w \leftarrow \exp_w(\mu \nabla_w f) \tag{2.27}$$

$$\exp_w(\mu \nabla_w f) = w \cdot \cos ||\mu \nabla_w f|| + \frac{\mu \nabla_w f \cdot \sin ||\mu \nabla_w f||}{||\mu \nabla_w f||} \tag{2.28}$$

$$\nabla_w f = \frac{1}{n^2} \left(I - ww^T\right) Z \cdot \tanh\left(P^T\right) = \frac{1}{n^2} \left(Z - wP\right) \tanh^T(P) \tag{2.29}$$

Si noti che ww^T è una matrice $l \times l$, Z è una matrice $l \times n^2$ e P^T è un vettore $n^2 \times 1$, mentre il gradiente $\nabla_w f$ è un vettore $l \times 1$, pertanto le dimensione dell'espressione (2.29) sono consistenti. Inoltre la Formula (2.27) di aggiornamento del vettore dei pesi w ci consente di aggiornare il vettore

sulla base di quello attuale e secondo la direzione data dalla formula (2.29) del gradiente $\nabla_w f$. La formula (2.29) può essere ricavata dalla formula del gradiente Euclideo della funzione f [3]:

$$f(w) := \mathcal{E}_z[R(w^T z)] = \int R(w^T z)\rho_z(z)dz^l, \tag{2.30}$$

dove R rappresenta una funzione non lineare stabilita da Hyvärinen e $\rho_z(z)$ la distribuzione di probabilità dei vettori z cioè degli ingressi della rete neurale ICA (dopo il centering e il whitening). Inoltre, si è utilizzato l'operatore di "valore atteso" (*expectation*) $\mathcal{E}_z[\cdot]$. Possiamo approssimare l'integrale come una sommatoria con k che da 1 arriva a n^2 dove n^2 è il numero di pixel dell'immagine:

$$f(w) \approx \sum_{k=1}^{n^2} R(w^T x_k)\Pr(z_k)\Delta z^l, \tag{2.31}$$

dove la quantità $\Pr(z_k)$ rappresenta la probabilità di incontrare il campione z_k e l'espressione Δz^l rappresenta il volume di un "iper-cubetto" centrato intorno al punto z_k. Calcolando il gradiente Euclideo della funzione $f(w)$ rispetto al vettore w, si ottiene:

$$\frac{\partial f}{\partial w} = \sum_{k=1}^{n^2} r(w^T z_k) z_k \Pr(z_k)\Delta z^l, \tag{2.32}$$

dove la funzione $r := R'$ nella nostra applicazione è data da $r := \tanh$. Abbiamo quindi ricavato l'equazione (2.29). Si può vedere anche che, a meno di una costante additiva arbitraria, l'integrale di r risulta $R = \ln \circ \cosh$. Per semplicità, si può ipotizzare che i campioni z_k siano distribuiti uniformemente, dunque che $\Pr(z_k) = \frac{1}{n^2}$, ottenendo:

$$\frac{\partial f}{\partial w} = \frac{\Delta z^l}{n^2} \sum_{k=1}^{n^2} \tanh(w^T z_k) z_k. \tag{2.33}$$

Infine, conviene osservare che se si lascia che la funzione tanh agisca componente per componente su un vettore (*component-wise*), risulta:

$$\frac{1}{n^2} \sum_{k=1}^{n^2} \tanh(w^T z_k) z_k =$$

$$\frac{1}{n^2} \left(\tanh(w^T z_1) z_1 + \tanh(w^T z_2) z_2 + \tanh(w^T z_3) z_3 + \cdots + \tanh(w^T z_{n^2}) z_{n^2} \right) =$$

$$\frac{1}{n^2} Z \tanh(Z^T w) = \frac{1}{n^2} Z \tanh(P^T)$$

e, dunque:

$$\frac{\partial f}{\partial w} = \frac{\Delta z^l}{n^2} Z \tanh(P^T). \tag{2.34}$$

Dato che la quantità Δz^l risulta costante, può essere assorbita nel learning rate μ ed è stata, pertanto, ignorata in (2.29).

3. Risultati degli esperimenti.

3.1. Risultati degli esperimenti con l'immagine di prova. Abbiamo dato in input al nostro programma un'immagine di prova di dimensione 240×240 mostrata in Figura 3.1 (a1). Abbiamo creato due PSF:

- Gaussian(1,1) con $\theta_x = 1$ e $\theta_y = 1$ mostrati in Figura 3.1 (a3)
- Gaussian(2,2) con $\theta_x = 2$ e $\theta_y = 2$ mostrati in Figura 3.2 (b3)

Applicando il PSF all'immagine originale, otteniamo l'immagine sfocata mostrata in Figura 3.1 (a2) per quella creata con il PSF Gaussian (1,1) e in Figura 3.2 (b2) per quella con il PSF Gaussian (2,2). Applichiamo ora all'immagine sfocata i filtri di Gabor precedentemente mostrati in Figura 2.1 ottenendo così 16 immagini, le quali insieme all'immagine sfocata, vengono poste in input all'algoritmo ICA.

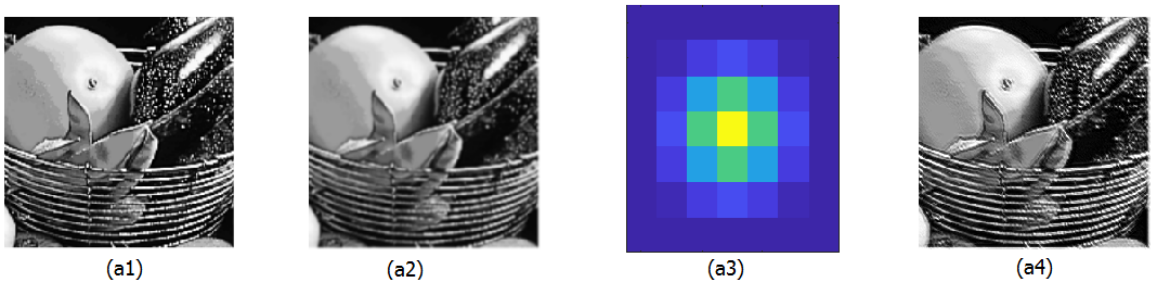

FIGURA 3.1. *Immagine originale (a1), sfocata (a2), PSF Gaussian(1,1) (a3), immagine ricostruita (a4). I risultati mostrati sono stati ricavati con l'algoritmo di apprendimento con i Manifold.*

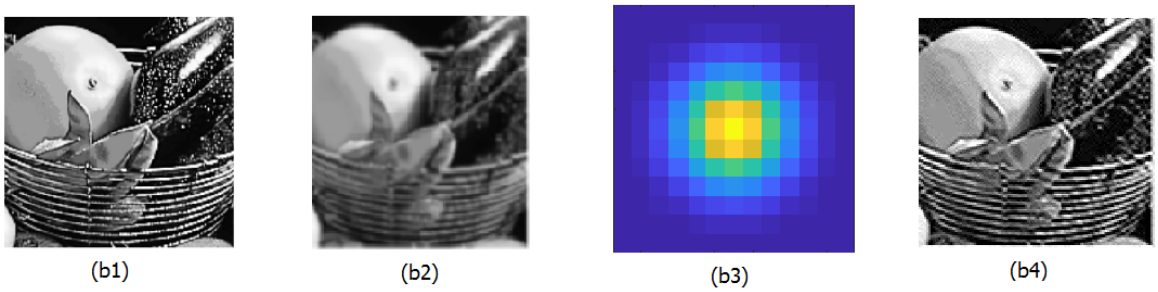

FIGURA 3.2. *Immagine originale (b1), sfocata (b2), PSF Gaussian(2,2) (b3), immagine ricostruita (b4). I risultati mostrati sono stati ricavati con l'algoritmo di apprendimento con i Manifold.*

L'algoritmo ICA ci ha restituito un vettore riga P di dimensione $1 \times n^2$ dove n^2 è il numero totale dei pixel di ciascuna immagine. Abbiamo convertito il vettore riga in una matrice $n \times n$ e abbiamo stampato la matrice come immagine. È possibile vedere il risultato delle due ricostruzioni in Figura 3.1 (a4) e in Figura 3.2 (b4). In questo esperimento dopo aver effettuato il centering e il whitening con il relativo restringimento, abbiamo ottenuto una matrice Z di 9 righe ($l = 9$).

PSF utilizzato	Originale/Sfocata	Originale/Restaurata (Umeyama)	Originale/Restaurata (Manifold)
Gaussian(1,1)	0.9658	0.9682	0.9684
Gaussian(2,2)	0.8836	0.9484	0.9509

TABELLA 3.1
Esempio dei valori di correlazione ottenuti con un'esecuzione del programma.

Per il funzionamento corretto dell'algoritmo ICA con i Manifold abbiamo avuto la necessità di impostare un *learning parameter* più piccolo affinchè si possa ottenere un risultato accettabile (10^{-6}) rispetto al metodo proposto da Umeyama il quale riesce a far convergere la rete neurale anche con un valore di 10^{-5}. Per i confronti tra i due metodi abbiamo deciso quindi di imporre il *learning parameter* come 0.00001.

Per mostrare la validità del metodo abbiamo calcolato i coefficienti di correlazione tra l'immagine originale e l'immagine sfocata, tra l'immagine originale e l'immagine ricostruita dall'ICA di Umeyama e infine tra l'immagine originale e l'immagine ricostruita dall'ICA con i Manifold mostrati nella Tabella 3.1. Notiamo che con i due metodi proposti otteniamo un buon risultato di ricostruzione dell'immagine. Si passa infatti da un valore 0.8836 a un valore 0.9509.

Dalla Tabella 3.1 si nota subito che il PSF che modifica maggiormente l'immagine il Gaussian(2,2) e che l'algoritmo che utilizza i concetti dei Manifold riesce a ricostruire meglio l'immagine rispetto all'algoritmo utilizzato da Umeyama. Inoltre, nel caso del PSF Gaussian(1,1), i coefficienti di correlazione tra i due metodi sono pressochè identici in quanto il PSF ha dimensioni molto ridotte rispetto all'immagine originale e quindi la sfocature risulta più lieve.

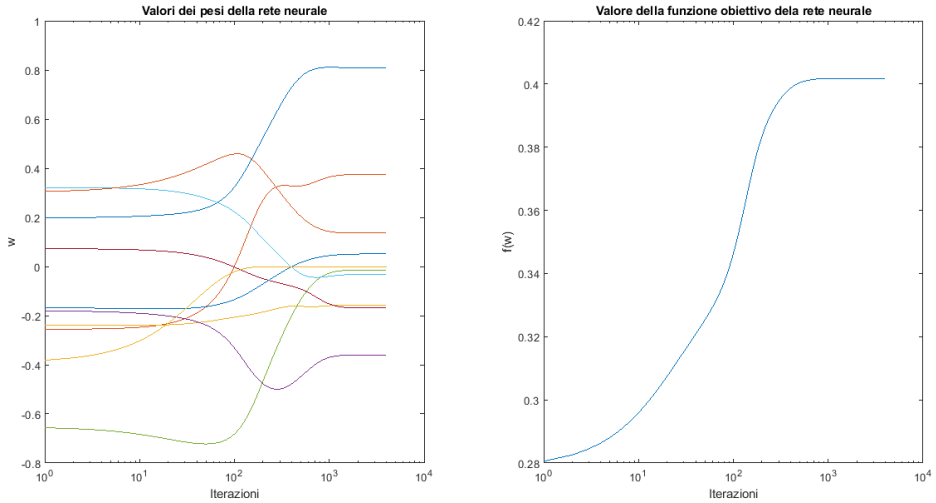

FIGURA 3.3. *Grafici di convergenza della rete neurale riferiti all'algoritmo con i Manifold.*

La Figura 3.3 mostra nel primo grafico la convergenza degli l pesi (9 in questo caso) del vettore w. Notiamo che il numero di iterazioni è sufficiente per completare l'operazione in quanto le varie curve convergono intorno a 1000 iterazioni a fronte delle 4000 predisposte. A lato è inoltre presente il grafico della convergenza della funzione obiettivo $f(w)$ durante l'apprendimento della rete

neurale ricavata nel Paragrafo 2.4 ed è ben visibile la sua convergenza. Possiamo quindi dedurre che il numero di operazioni per l'apprendimento della rete neurale è sufficiente.

Abbiamo, inoltre, calcolato i valori di correlazione medi eseguendo 100 volte il programma che esegue la ricostruzione con i due metodi. Abbiamo generato così 100 vettori dei pesi w i quali sono stati utilizzati dai due metodi, in modo da poter avere una media dei valori di correlazione da poter confrontare. I valori di correlazione medi sono mostrati nella Tabella 3.2.

PSF utilizzato	Originale/Sfocata	Originale/Restaurata (Umeyama)	Originale/Restaurata (Manifold)
Gaussian(1,1)	0.9658	0.9628	0.9630
Gaussian(2,2)	0.8836	0.7698	0.7728

TABELLA 3.2
Valori di correlazione medi calcolati con i PSF utilizzati.

Come è possibile notare, il valore di correlazione medio tra l'immagine originale e l'immagine restaurata 0.7698 è di gran lunga inferiore a quello tra l'immagine originale e sfocata 0.8836. Questo significa che i due metodi, nel caso medio, non riescono a migliorare l'immagine sfocata andandola addirittura a peggiorare. Questo problema è dovuto principalmente alla scelta del vettore dei pesi w, il quale viene generato casualmente nella prima iterazione per poi essere modificato durante l'apprendimento della rete neurale. In alcuni casi può capitare che il vettore dei pesi w iniziale, essendo casuale, capiti in un'area dove la funzione da massimizzare è piatta (il gradiente è quasi zero) e si arresta oppure il numero di iterazioni predisposte dall'algoritmo (4000 nel nostro caso) non sono sufficienti per ricostruire l'immagine.

È possibile vedere un esempio di risultato non ottimale del nostro algoritmo nella Figura 3.4: si è passati da un valore di 0.8836 ad un valore di 0.7746 .

FIGURA 3.4. *Esempio di ricostruzione non completata correttamente.*

Per concludere, abbiamo calcolato la convergenza media della rete neurale di entrambi gli algoritmi (eseguendoli 100 volte), in modo da confrontare la velocità della loro convergenza. Dalla Figura 3.5 possiamo notare come la rete neurale con l'algoritmo di apprendimento con i Manifold converge prima dell'algoritmo proposto da Umeyama. Questo ci porta a dedurre che la convergenza con il primo metodo risulta più veloce del secondo.

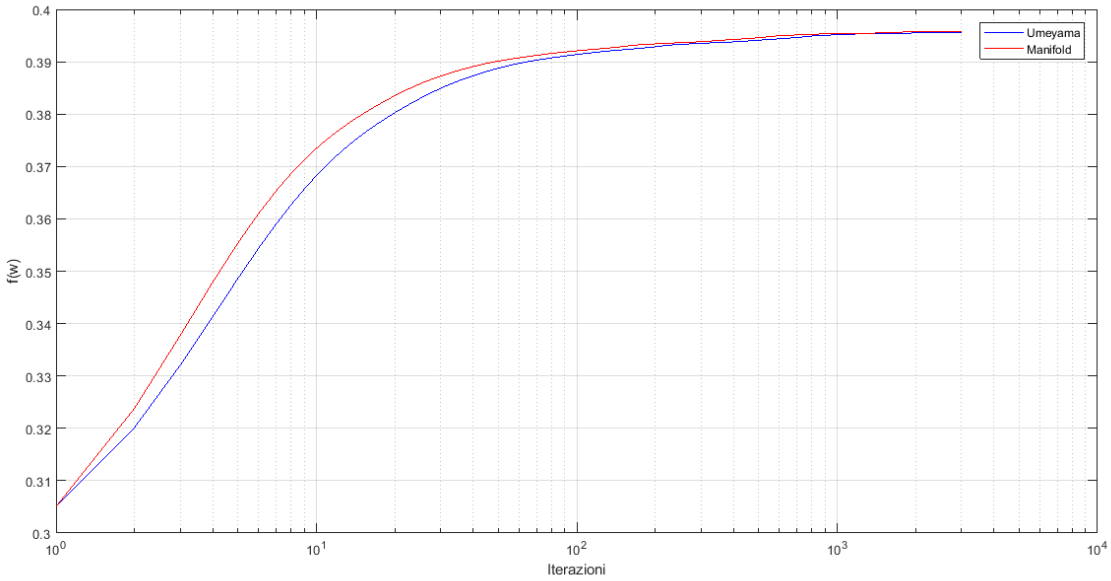

FIGURA 3.5. *Grafico della convergenza della rete neurale nei due algoritmi studiati, mediato su 100 test indipendenti.*

3.2. Casistiche in cui il metodo non funziona. Il metodo proposto, inoltre, non è efficiente per tutte le varie tipologie di immagini in quanto se si va ad utilizzarne alcune che contengono oggetti posti a diverse distanze, come per esempio un paesaggio, il metodo non riesce a ricostruire perfettamente l'immagine andandola addirittura in alcuni casi a peggiorare come possibile vedere in Figura 3.6.

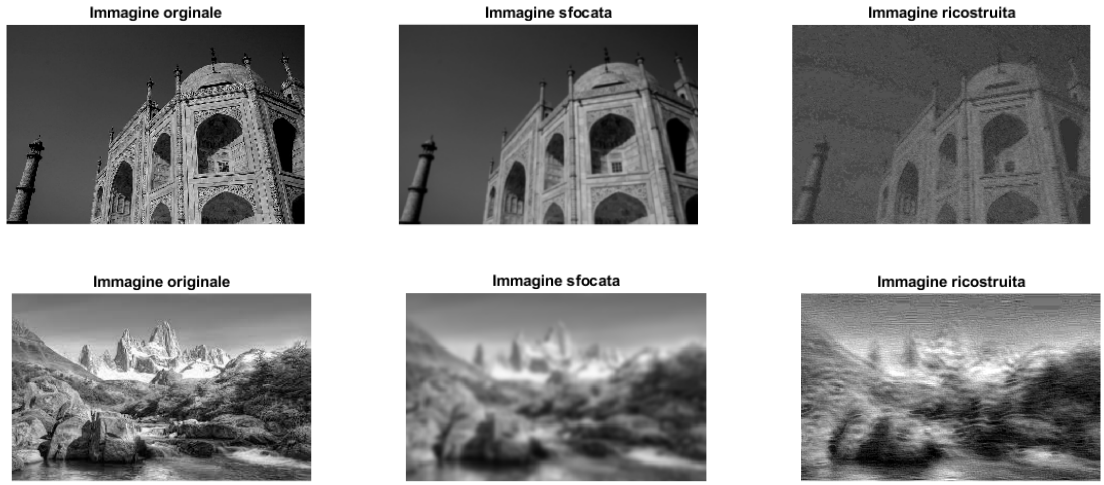

FIGURA 3.6. *Le immagini di un monumento e di un paesaggio non vengono correttamente ricostruite dal metodo.*

3.3. Ripristino di un'immagine realmente sfocata. Per un'ulteriore verifica della correttezza del procedimento abbiamo applicato i due metodi a un'immagine realmente sfocata (senza quindi utilizzare il PSF). Di seguito è mostrata una foto di alcuni libri scattata con una macchina fotografica senza essere messa a fuoco. Abbiamo processato l'immagine con i 16 filtri di Gabor e l'output dei filtri, insieme all'immagine originale, viene dato in input all'ICA. L'immagine restaurata con i due metodi è mostrata nella Figura 3.7. Come si pu vedere, i nomi dei libri che prima non erano distinguibili sono ora leggibili come ad esempio il nome dell'autore del libro centrale *Dan Brown*. È importante sottolineare che per utilizzare questo metodo serve che il PSF sia uniforme su tutta l'immagine e per questo motivo abbiamo fotografato dei libri con la stessa profondità, in modo da non incorrere nei problemi enunciati nella Sezione 3.2. Si è dimostrato quindi che i due metodi possono essere ulteriormente studiati ed approfonditi per future applicazione reali.

FIGURA 3.7. *Immagine realmente sfocata ricostruita dai metodi studiati. L'immagine mostra rispettivamente la foto sfocata (a1), la foto ricostruita dall'algoritmo di Umeyama (a2) e la foto ricostruita dall'algoritmo con i Manifold (a3).*

Per completezza, nella Figura 3.8 mostriamo che il risultato della ricostruzione di un'immagine reale ripresa non frontalmente non è soddisfacente.

FIGURA 3.8. *Immagine realmente sfocata ricostruita dai metodi studiati. L'immagine mostra rispettivamente la foto sfocata (b1), la foto ricostruita dall'algoritmo di Umeyama (b2) e la foto ricostruita dall'algoritmo con i Manifold (b3).*

4. Conclusioni. In questa relazione abbiamo illustrato il procedimento per la ricostruzione delle immagini proposto da Umeyama e abbiamo fornito una variante con i Manifold utilizzando le conoscenze dell'analisi numerica applicate alle reti neurali e all'*image processing*, in particolare alcune nozioni riguardanti la mappa esponenziale e la sfera unitaria. I due metodi proposti possono ricostruire l'immagine senza conoscere il PSF presente nel modello di sfocatura.

Abbiamo dimostrato nel documento che i metodi riescono a funzionare correttamente ottenendo sotto determinate ipotesi anche un livello di correlazione alto tra l'immagine originale e

l'immagine ricostruita, soffermandoci anche sulle motivazioni per cui non si riesce ad ottenere i risultati sperati. Abbiamo dimostrato come la convergenza dell'algoritmo ICA con i Manifold sia più veloce di quella proposta da Umeyama ma allo stesso tempo necessita di una precisione maggiore.

Abbiamo infine fornito un esempio di un'applicazione reale dell'algoritmo ICA, inserendo in input un'immagine già sfocata, notando che se questa viene ripresa frontalmente otteniamo buoni risultati, mentre negli altri casi il risultato non soddisfacente.

RIFERIMENTI BIBLIOGRAFICI

[1] Shinji Umeyama, Blind Deconvolution of Images using Gabor Filters and Independent Component Analysis, ICA2003, Nara, Japan.

[2] Shinji Umeyama, Blind deconvolution of Blurred Images by Use of ICA, Electronics and Communications in Japan, Part 3, Vol.84, No. 12,2001.

[3] Aapo Hyvrinen and Erkki Oja, Independent Component Analysis: A Tutorial, April 1999.

[4] Simone Fiori, On vector averaging over the unit hypersphere, 17 July 2008.

[5] Simone Fiori, Introduction to Curved Spaces (Manifolds), Lecture 1: Smooth manifolds, curves and tangent spaces.