Alla mia famiglia

Ringraziamenti

Vorrei ringraziare il Prof. Ivan Gerace che incontrai quando ero ancora uno studente del corso di laurea triennale in informatica. Dopo avermi introdotto nell'area dell'"image processing", mi ha sempre spinto verso nuove sfide. Fin dall'inizio mi ha dato la possibilità di approfondire e sviluppare la mia passione verso questo campo di ricerca. Mi ha anche fatto conoscere Francesco Petrogalli con cui ho preparato buona parte di questo lavoro. Sono grato a Francesco per la sua disponibilità e il suo interesse verso i problemi che si sono presentati nello sviluppo di questa tesi.

Desidero inoltre ringraziare il Prof. Marco Baioletti per i vari consigli dati sull'ottimizzazione a sciame di particelle. Un grazie anche a mio fratello Gabriele per il supporto datomi in questo periodo.

Indice

Elenco delle figure

Introduzione

Una delle prime macchine in grado di tradurre una foto in un codice digitale fu il sistema di trasmissione Bartlane che prende il nome dai sui inventori Mr. Bartholomew e Captain MacLane. Costruito intorno agli anni '20, questo consentiva di spedire foto da Londra a New York, con una qualità sufficiente da renderle stampabili sui giornali. Le immagini venivano codificate su un nastro perforato (Baudot Tape) e poi trasmesse su un canale telegrafico. Si cercò fin da subito di migliorare la qualità delle immagini inviate passando da 5 livelli di grigio codificati nel 1921 a 15 nel 1929.

L'elaborazione mediante computer iniziò negli anni '60 quando vennero lanciate le prime sonde lunari. In quegli anni i ricercatori del Jet Propulsion Laboratory della NASA furono in grado di trasformare le immagini corrotte e distorte provenienti dalle sonde Ranger e Surveyer in foto dettagliate che finirono sulle pagine della rivista "Life". Da allora i campi di applicazione sono cresciuti enormemente a tal punto da coinvolgere molti settori della ricerca scientifica. Tra i più importanti si menzionano il campo medico, astronomico e archeologico. Oggi in queste branche le immagini costituiscono un fondamentale strumento di analisi, ma nonostante i continui progressi nelle tecniche di acquisizione, molto spesso è impossibile utilizzarle direttamente a causa di distorsioni dovute a sfocatura e rumore. È pertanto necessario cercare di rimuovere l'effetto di queste degradazioni.

Quando si conoscono le caratteristiche della sfocatura e del rumore, il problema che cerca la migliore stima dell'immagine originale partendo da una sua forma degradata è chiamato problema del *restauro d'immagine*. Questo è malposto *nel senso di Hadamard* [BPT88] ovvero la soluzione può non esistere, non essere unica, non essere stabile rispetto a variazioni sui dati. Per rendere il problema trattabile, sono necessarie delle tecniche di regolarizzazione [Dem89, Reg96] che ne ristabiliscono la natura ben-posta.

La soluzione del problema del restauro d'immagine può essere definita come

il minimo di una specifica funzione (*funzione energia*) data dalla somma di due termini. Il primo quantifica la fedeltà con i dati, mentre il secondo misura l'"uniformità" della soluzione. La funzione energia non è convessa e la sua minimizzazione, in questa tesi, è data da un algoritmo deterministico di tipo GNC (*Graduated Non-Convexity*) [bz87, BGT94, Nik99].

Sfortunatamente in molti casi il processo di degradazione è sconosciuto. In questa situazione il problema si complica perchè insieme all'immagine originale bisogna stimare anche le caratteristiche della sfocatura e del rumore. Nel presente lavoro di tesi questo problema, noto come *restauro cieco d'immagine*, è stato affrontato utilizzando un metodo basato sull'ottimizzazione a sciame di particelle [KE95, SE98, vdBE06]. L'idea base è che la posizione di ogni particella identifica un operatore di sfocatura e che la bontà (funzione di fitness) di ciascuno di questi può essere determinata valutando la funzione energia sul risultato del restauro eseguito con l'operatore stesso. Con questa formulazione, il problema può essere risolto facendo evolvere la popolazione verso il minimo della funzione obiettivo. Per migliorare la complessità dell'algoritmo la funzione di fitness viene valutata localmente in una sottoimmagine più piccola di quella osservata. È chiaro che l'informazione fornita al processo di ottimizzazione dipende dalle caratteristiche della zona considerata. Pertanto il processo di estrazione richiede che la "regolarità" della sottoimmagine, cada in un intervallo prefissato. Se questo è definito in modo adeguato, allora ogni sottoimmagine, presa a caso dai dati, sarà ugualmente "significativa". Infine, il processo di estrazione, dopo un numero fissato di "movimenti", viene ripetuto, reinizializzando di volta in volta la posizione di tutte le particelle tranne una che mantiene la migliore trovata al passo precedente.

La struttura di questa tesi è organizzata come segue. Nel Capitolo 1 si introducono le degradazione trattate, il formato di immagine utilizzato e il modello matematico del processo di degradazione. Nel Capitolo 2 si descrive il problema del restauro d'immagine sia quando si conosce il processo di degradazione, sia quando questo è sconosciuto. Nel Capitolo 3 viene fornita una panoramica dell'ottimizzazione a sciame di particelle. Il Capitolo 4 illustra le caratteristiche dell'algoritmo proposto per risolvere il problema del restauro cieco. Infine nel Capitolo 5 si riportano e si discutono i risultati sperimentali.

1

Concetti base

Un'immagine può essere definita come una funzione bidimensionale, $f(x, y)$, dove x e y sono le coordinate spaziali, e il valore di f per ogni coppia (x, y) corrisponde all'intensità luminosa –e quindi al colore– dell'immagine in quel punto. La digitalizzazione di un'immagine presuppone che x, y, e l'intensità f siano tutte quantità finite e discrete. Ogni immagine digitale può quindi essere vista come un numero finito di elementi, ciascuno con una propia posizione e un propio valore. Questi sono chiamati "*picture elements*" o più semplicemente "*pixels*".

Nonostante i continui progressi nelle tecniche di digitalizzazione, molto spesso durante l'acquisizione da fotocamere, videofonini, scanner, microscopi, etc... vengono introdotti disturbi che degradano la qualità dell'immagine. Lo scopo del *restauro d'immagine* è quello di, partendo da un'immagine degradata, riportarla ad una stima quanto più vicina possibile a quella dell'immagine originale, cioè priva di disturbi.

In questo capitolo si introduce il formato di immagine utilizzato e le degradazione trattate.

1.1 Formato d'immagine

Per rappresentare e manipolare le immagini è stato utilizzato il formato PGM (Portable Grey Map). Come intuibile dall'acronimo, viene utilizzata una scala di grigi che fa corrispondere il valore di ogni pixel ad una tonalità (*grey level*). In particolare ogni pixel è codificato con 8-bit per un totale di 256 tonalità di grigio specificate da valori interi che vanno da 0 (nero) a 255 (bianco).

Ogni immagine PGM ha la seguente struttura definita in un file ASCII:

- Un "numero magico" per identificare il tipo di file. Nel caso specifico tale numero magico è denotato da due caratteri: "P2".
- Spazio bianco (blanks, TABs, CRs, LFs).
- Una larghezza, formattata come un carattere decimale (corrisponde al numero di colonne della matrice che rappresenta l'immagine).
- Spazio bianco.
- Un'altezza, ancora formattata come un carattere decimale (corrisponde al numero di righe della matrice che rappresenta l'immagine).
- Spazio bianco.
- Un carattere decimale che specifica il numero di tonalità di grigio.
- Spazio bianco.
- Una matrice (larghezza $\times$ altezza) di tonalità di grigio, con valori compresi tra 0 e 255, separati da uno spazio bianco. Ogni elemento della matrice definisce il valore del corrispondente pixel dell'immagine.

Grazie alla sua semplicità, che consente di scrivere applicativi in maniera diretta, il formato PGM è molto utilizzato nella ricerca relativa all'analisi e alla manipolazione di immagini.

1.2 Degradazioni trattate

Le degradazioni più comuni introdotte durante l'acquisizione di un'immagine sono la sfocatura e il rumore. La prima è una distorsione deterministica che si manifesta attraverso la non chiara distinzione dei bordi degli oggetti, rappresentati nell'immagine. La seconda è una perturbazione probabilistica che altera il reale valore dei pixel in funzione di una distribuzione di probabilità. In questa tesi si assume che il rumore sia additivo, bianco, gaussiano, con media zero e varianza σ^2 nota.

Matematicamente la sfocatura può essere ottenuta sostituendo il valore di un particolare pixel con la media pesata dei valori dei pixel adiacenti. I pesi coinvolti possono essere rappresentati come le entrate di una matrice $M \in \mathbb{R}^{(2h+1)\times(2h+1)}$, $h \in \mathbb{N}^+$, detta *maschera di blur*.

Consideriamo ad esempio l'immagine $\mathbf{x}$ in forma matriciale di dimensione 5×5 e la maschera M di dimensione 3×3:

$$\mathbf{x} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} \\ x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} \end{pmatrix} \qquad M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

L'effetto dell'applicazione della maschera ad ogni pixel di $\mathbf{x}$ è il seguente:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{x}_{1,1} & \hat{x}_{1,2} & \hat{x}_{1,3} & \hat{x}_{1,4} & \hat{x}_{1,5} \\ \hat{x}_{2,1} & \hat{x}_{2,2} & \hat{x}_{2,3} & \hat{x}_{2,4} & \hat{x}_{2,5} \\ \hat{x}_{3,1} & \hat{x}_{3,2} & \hat{x}_{3,3} & \hat{x}_{3,4} & \hat{x}_{3,5} \\ \hat{x}_{4,1} & \hat{x}_{4,2} & \hat{x}_{4,3} & \hat{x}_{4,4} & \hat{x}_{4,5} \\ \hat{x}_{5,1} & \hat{x}_{5,2} & \hat{x}_{5,3} & \hat{x}_{5,4} & \hat{x}_{5,5} \end{pmatrix}.$$

In particolare il pixel $\hat{x}_{3,4}$ è calcolato come segue:

$$\hat{x}_{3,4} = \frac{1}{9}\,(x_{2,3}\ x_{2,4}\ x_{2,5}\ x_{3,3}\ x_{3,4}\ x_{3,5}\ x_{4,3}\ x_{4,4}\ x_{4,5}) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Sui bordi e sugli spigoli, la maschera viene applicata con un numero infe-

riore di punti. Per esempio il pixel $x_{b(1,5)}$ è calcolato come:

$$\hat{x}_{1,5} = \frac{1}{4}\,(x_{1,4}\ x_{1,5}\ x_{2,4}\ x_{2,5}) \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix}.$$

Formalmente si assume che l'immagine osservata $\vec{\mathbf{y}}$ derivi da un'immagine ideale $\vec{\mathbf{x}}$, sfocata da un operatore A e corrotta da un rumore aggiuntivo $\vec{\mathbf{n}}$.

$$\vec{\mathbf{y}} = A\vec{\mathbf{x}} + \vec{\mathbf{n}}, \tag{1.1}$$

dove $A \in \mathbb{R}^{N^2 \times N^2}$ è un operatore lineare, $\vec{\mathbf{y}}, \vec{\mathbf{x}}, \vec{\mathbf{n}} \in \mathbb{R}^{N^2}$ sono vettori in forma lessicografica[1] e $N^2 \in \mathbb{N}^+$ è il numero di pixel dell'immagine. L'operatore A nell'equazione (1.1) definisce la sfocatura introdotta e può essere ricavato dalla maschera di blur M come segue:

$$a_{(i,j)(i+w,j+v)} = \begin{cases} \dfrac{m_{h+1+w,h+1+v}}{v_{i,j}} & \text{se } |w|, |v| \leq h, \\ 0 & \text{altrimenti,} \end{cases}$$

dove

$$v_{i,j} = \sum_{k=b}^{c} \sum_{l=d}^{e} m_{k,l},$$

$$\begin{aligned} b &= \max\{1, 2+h-i\}, \quad & c &= \min\{2h+1, N-i+h+1\}, \\ d &= \max\{1, 2+h-j\}, \quad & e &= \min\{2h+1, N-j+h+1\}. \end{aligned}$$

[1] La forma lessicografica si ottiene ordinando le colonne di una matrice in un vettore.

2

Il problema del restauro d'immagine

Nel Capitolo 1 si è accennato al problema del restauro come il problema di ritrovare l'immagine originale, partendo da una sua forma degradata. In questo capitolo vedremo come risolvere tale problema, noto come *problema inverso*, quando si conosce l'operatore di sfocatura A (Paragrafo 2.1 e 2.2), e come affrontarlo quando invece il processo di degradazione è sconosciuto (Paragrafo 2.3).

2.1 Problema inverso

Il modello matematico che definisce il problema inverso è identico a quello dell'equazione (1.1). In questo caso tuttavia l'obiettivo è quello di ricavare l'immagine ideale $\vec{\mathbf{x}}$ conoscendo l'immagine degradata $\vec{\mathbf{y}}$, la maschera di blur M e la varianza σ^2 del rumore. Risolvere questo problema non è banale poichè malposto *nel senso di Hadamard* [BPT88]. Infatti la soluzione può non esistere, non essere unica, non essere stabile rispetto a variazioni sui dati. Per rendere il problema trattabile è necessario utilizzare una *tecnica di regolarizzazione* [Dem89, Reg96], che impone vincoli di uniformità sulla soluzione.

Un modello realistico di immagine tuttavia alterna regioni regolari, caratterizzate dalla stessa intensità luminosa, a zone di interruzione che corrispondono ai bordi degli oggetti. I bordi identificano discontinuità che possono andare perse in seguito all'imposizione dei vincoli. Per evitare che ciò accada è ne-

cessario introdurre delle variabili ausiliarie, chiamate *variabili di linea*, che permettano di interrompere la regolarità la dove è presente una discontinuità. Per preservare quest'ultime sia sull'immagine trattata che sulla sua derivata prima, sono stati proposti in letteratura diversi metodi che sfruttano *differenze finite del secondo ordine* [GPPS08].

Definizione. Una variabile di linea b_c è una variabile booleana associata ad una *clique* c. Se nella clique c è presente una discontinuità, allora la corrispondente variabile di linea b_c assumerà il valore uno, zero altrimenti. L'insieme $\vec{\mathbf{b}}$ di tutte le variabili di linea, è chiamato *processo di linea*.

Definizione. Sia $Q_n = ([1,n] \cap \mathbb{N}) \times ([1,n] \cap \mathbb{N})$ per ogni $n \in \mathbb{N}$. Una clique di ordine k è l'insieme dei punti di Q_n in cui risultano definite le differenze finite di ordine k.

Indichiamo con il simbolo C_k l'insieme di tutte le clique di ordine k cosicchè,

$$C_1 = \{c = \{(i,j),(h,l)\} : i = h, j = l+1 \text{ oppure } i = h+1, j = l\},$$

è l'insieme di tutte le clique del primo ordine e

$$\begin{aligned} C_2 \;=\; & \{c = \{(i,j),(h,l),(r,q)\} : i = h = r, j = l+1 = q+2 \\ & \text{oppure } i = h+1 = r+2, j = l = q\}, \end{aligned}$$

l'insieme di tutte quelle del secondo ordine.

Indichiamo inoltre con $D_c^k\vec{\mathbf{x}}$, l'operatore differenza finito del vettore $\vec{\mathbf{x}}$ sulla clique c di ordine k. In particolare se $c = \{(i,j),(h,l)\} \in C_1$

$$D_c^1\vec{\mathbf{x}} = x_{i,j} - x_{h,l},$$

se $c = \{(i,j),(h,l),(r,q)\} \in C_2$

$$D_c^2\vec{\mathbf{x}} = x_{i,j} - 2x_{h,l} + x_{r,q}.$$

Ora, fissati $\vec{\mathbf{x}}$ e k, se si verifica che il valore di $D_c^k\vec{\mathbf{x}}$, in una determinata clique c, supera una certa soglia, significa che in c è presente una disconti-

nuità, altrimenti c identifica una zona regolare. Per fare in modo che questa informazione non vada persa, gli si associa la corrispondente variabile di linea.

2.1.1 Definizione della funzione energia

La soluzione che si cerca, oltre che preservare le discontinuità, deve anche essere coerente con i dati conosciuti a priori. In altre parole, conoscendo la maschera di blur M e la varianza σ^2 del rumore bisogna modellare la soluzione $\vec{\mathbf{x}}$, in modo tale che il processo di degradazione applicato a $\vec{\mathbf{x}}$ fornisca un risultato il più simile possibile all'immagine degradata $\vec{\mathbf{y}}$. Formalmente si può quantificare la fedeltà della soluzione con i dati come:

$$||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2, \tag{2.1}$$

dove $||\cdot||$ indica la norma euclidea. Il vincolo di regolarità accennato in precedenza, può invece essere espresso come:

$$\sum_{c \in C_k} \left[\lambda^2 (D_c^k \vec{\mathbf{x}})^2 (1 - b_c) + \alpha b_c\right]. \tag{2.2}$$

Si può notare che ogni elemento della sommatoria, dipende dalla clique c considerata e vale o α o $\lambda^2 (D_c^k \vec{\mathbf{x}})^2$ a seconda che c identifichi rispettivamente nell'immagine $\vec{\mathbf{x}}$, una discontinuità o una zona uniforme. La variabile reale λ, chiamata *parametro di regolarizzazione*, è utilizzata per ottimizzare l'uniformità della soluzione. Valori di λ vicini allo zero consentono di trovare una soluzione più ricca di particolari e fedele ai dati. Aumentando λ invece si ottiene una maggiore regolarità nelle scene dell'immagine accompagnata da una perdita di dettagli. Infine α corrisponde a un peso associato ad ogni variabile di linea e può essere visto come un costo da pagare in caso di discontinuità.

Sommando la (2.1) e la (2.2) si ottiene il seguente funzionale:

$$E(A, \vec{\mathbf{x}}, \vec{\mathbf{b}}) = ||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \sum_{c \in C_k} \left[\lambda^2 (D_c^k \vec{\mathbf{x}})^2 (1 - b_c) + \alpha b_c\right]. \tag{2.3}$$

La (2.3) viene chiamata *funzione energia primale* e definisce una quantità

positiva, in quanto ambedue i termini che la compongono sono positivi.

Il termine (2.1) è tanto meno influente quanto più l'immagine è fedele con i dati. Letteralmente più piccola è la differenza –e quindi la stima fedele ai dati– meno, la (2.1), "pesa" nella (2.3). Analogamente il termine (2.2) è tanto meno influente quanto più l'immagine è regolare. Infatti, come detto in precedenza, ogni clique c contribuirà nella (2.3) con il valore α o $\lambda^2(D_c^k\vec{\mathbf{x}})^2$. Nel primo caso il contributo è dato da un costo fisso dato da α. Nel secondo più l'immagine è regolare –e quindi il valore dei pixel coinvolti nella clique simile– più il valore $D_c^k\vec{\mathbf{x}}$ è piccolo.

Ora, poichè si cerca una stima che sia fedele ai dati il più regolare possibile, la soluzione del problema inverso può essere definita come la ricerca di quella coppia $(\vec{\mathbf{x}}, \vec{\mathbf{b}})$ che minimizza la (2.3).

2.1.2 Eliminazione del processo di linea

È possibile riformulare la (2.3) in modo da trattare le variabili di linea in maniera implicita [bz87, GR92]. Per fare questo bisogna prima riscriverla come:

$$E(A, \vec{\mathbf{x}}, \vec{\mathbf{b}}) = ||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \sum_{c \in C_k} h_{\alpha,\lambda}(D_c^k\vec{\mathbf{x}}, b_c), \tag{2.4}$$

dove

$$h_{\alpha,\lambda}(t, b_c) = \lambda^2 t^2(1 - b_c) + \alpha b_c. \tag{2.5}$$

In questo modo è evidente che la dipendenza di E dalle variabili di linea b_c è limitata alle n copie di $h_{\alpha,\lambda}$ nella (2.4), con n uguale alla cardinalità delle clique c. Da notare che $h_{\alpha,\lambda}$ vale o α o $\lambda^2 t^2$ a seconda che la variabile booleana b_c sia rispettivamente 1 o 0 (vedi Figura 2.1). Dal momento che l'obiettivo è minimizzare E, il problema può essere definito come:

$$\min_{(\vec{\mathbf{x}}, \vec{\mathbf{b}})} \left(||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \sum_{c \in C_k} h_{\alpha,\lambda}(D_c^k\vec{\mathbf{x}}, b_c) \right), \tag{2.6}$$

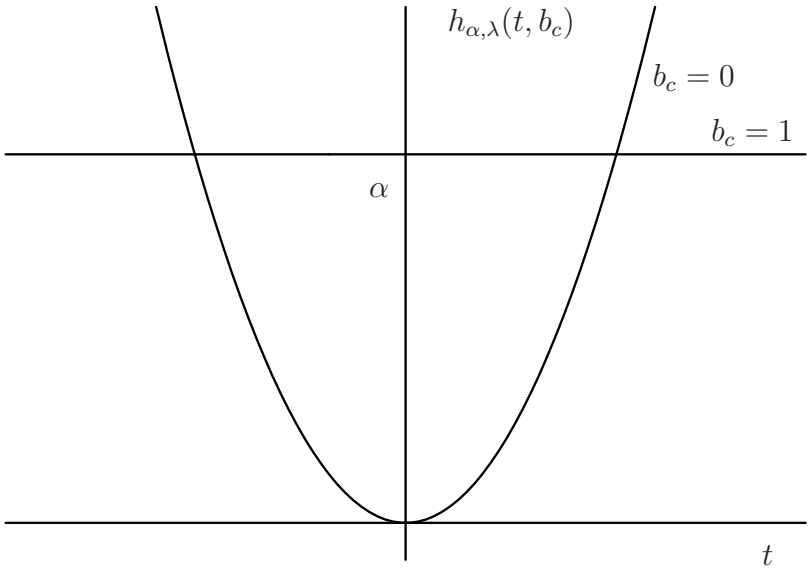


Figura 2.1: Funzione $h_{\alpha,\lambda}(t, b_c)$ per $\alpha = 10$ $\lambda = 0.6$.

ma poichè il termine $||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2$ non dipende da b_c, la (2.6) equivale a

$$\min_{\vec{\mathbf{x}}} \left(||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \min_{\vec{\mathbf{b}}} \sum_{c \in C_k} h_{\alpha,\lambda}(D_c^k \vec{\mathbf{x}}, b_c) \right).$$

Quindi se si minimizza prima rispetto a $\vec{\mathbf{b}}$ rimane il problema di minimizzare E_d rispetto $\vec{\mathbf{x}}$ dove

$$E_d(A, \vec{\mathbf{x}}) = ||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \sum_{c \in C_k} g_{\alpha,\lambda}(D_c^k \vec{\mathbf{x}}), \tag{2.7}$$

e

$$g_{\alpha,\lambda}(t) = \min_{b_c \in \{0,1\}} h_{\alpha,\lambda}(t, b_c) = \begin{cases} \lambda^2 t^2 & \text{se } \mid t \mid < \frac{\sqrt{\alpha}}{\lambda}, \\ \alpha & \text{altrimenti.} \end{cases} \tag{2.8}$$

La (2.7) è chiamata *funzione energia duale*, mentre la (2.8), che corrisponde al minimo dei due grafici in Figura 2.1, è chiamata *funzione di interazione*. La quantità $s = \frac{\sqrt{\alpha}}{\lambda}$ definisce la soglia entro cui ricercare le discontinuità.

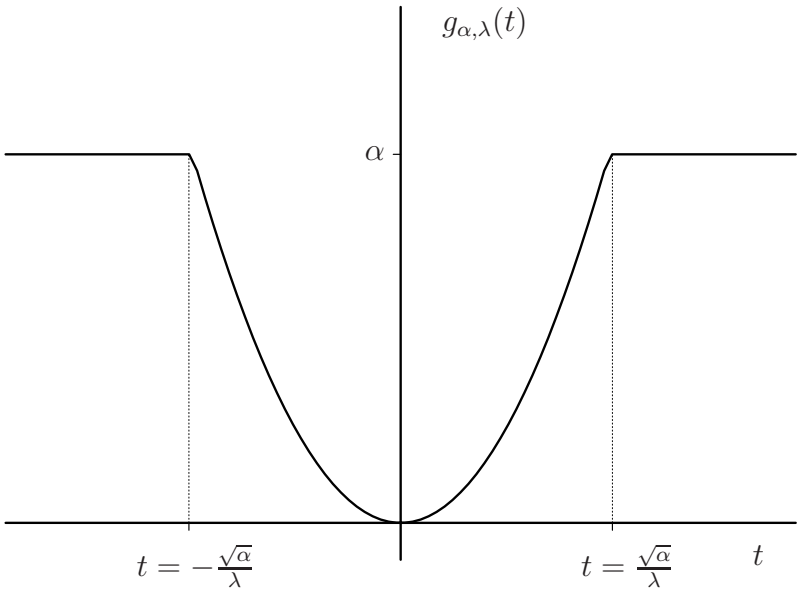


Figura 2.2: Funzione $g_{\alpha,\lambda}(t)$ per $\alpha = 10$ $\lambda = 0.6$.

La variabile di linea b_c, associata ad ogni clique c può essere facilmente recuperata:

$$b_c = \begin{cases} 0 & \text{se } \mid t \mid < \frac{\sqrt{\alpha}}{\lambda}, \\ 1 & \text{altrimenti.} \end{cases}$$

2.2 Algoritmi di minimizzazione

Minimizzare la funzione energia duale E_d –generalmente non convessa– non è un problema banale. In letteratura sono stati proposti sia algoritmi stocastici [SBP83, KGV83, GG84] che deterministici [bz87, Nik99, BGT94].

2.2.1 Simulated annealing

Il *simulated annealing* è una tecnica di ottimizzazione stocastica, fondata sull'algoritmo di Metropolis [MRR$^+$53], che consente di trovare l'ottimo globale di funzioni caratterizzate da molti ottimi locali. A differenza degli algoritmi di discesa classici che si muovo lungo la direzione del gradiente, nel simula-

ted annealing, una certa randomicità viene introdotta nel cammino di discesa. Questa consente letteralmente di "saltare fuori" dai minimi locali. Il grado di randomicità viene controllato da un parametro *temperatura* opportunamente decrementato man mano che ci si avvicina all'ottimo globale.

Uno schema generale di come una procedura basata sul simulated annealing possa essere applicata al restauro d'immagine è mostrato nell'algoritmo 2.1.

Algoritmo 2.1 Simulated Annealing

Require: $\vec{\mathbf{y}}$ l'immagine osservata
Require: T_0 una temperatura iniziale
Require: A l'operatore di sfocatura

```
x⃗ ← y⃗
T ← T_0
while T ≠ 0 do
    for all i : i ∈ ℕ, i ∈ [1, n] do
        r_1 ← pseudo-casuale da una distribuzione di probabilità
        ΔE_d ← E_d(A, x_1, ..., x_i, ..., x_n) − E_d(A, x_1, ..., x_i + r_1, ..., x_n)
        if ΔE_d ≥ 0 then
            x_i ← x_i + r_1
        else
            r_2 ← pseudo-casuale ∈ [0, 1]
            if r_2 ≤ e^(ΔE_d / T) then
                x_i ← x_i + r_1
            end if
        end if
    end for
    T ← T − γ {γ definisce lo schema di raffreddamento}
end while
```

In particolare una temperatura T viene gradualmente ridotta a partire da un suo valore iniziale T_0. Ad ogni passo una perturbazione casuale viene applicata ai pixel dell'immagine $\vec{\mathbf{x}}$, accettando quei cambiamenti che corrispondono ad una riduzione dell'energia. Formalmente preso un pixel x_i si calcola

$$\Delta E_d = E_d(A, x_1, \ldots, x_i, \ldots, x_n) - E_d(A, x_1, \ldots, x_i + \delta, \ldots, x_n),$$

dove δ indica la perturbazione introdotta.

Se $\Delta E_d \geq 0$ il cambiamento viene accettato, se invece $\Delta E_d < 0$ la modifica viene accettata con probabilità

$$e^{\frac{\Delta E_d}{T}}.$$

Quando T è "grande", perturbazioni che incrementano il valore dell'energia sono spesso accettate, consentendo al sistema di uscire da minimi locali. Quando $T \to 0$, il sistema si "raffredda", diventando quasi deterministico nella sua discesa verso il minimo.

2.2.2 Algoritmi deterministici di tipo GNC

Gli algoritmi stocastici, utilizzando perturbazioni casuali, sono meno meno efficienti di quelli deterministici [Bla89]. Il risultato di questi ultimi tuttavia, dipende fortemente dalla scelta del punto iniziale con cui vengono avviati. Per permettere alla minimizzazione di partire da un punto che si stima essere vicino al minimo globale si utilizza una tecnica nota come GNC (*Graduated Non-Convexity*) [bz87, Nik99, BGT94]. Questa si basa sulla costruzione di una famiglia di funzioni approssimanti $\left\{E_d^{(p)}\right\}_p$ –governate da un parametro p ($P \in \mathbb{R}, p \in (0,2]$)– della funzione energia duale E_d, di cui la prima, $E_d^{(2)}$, è convessa e l'ultima, $E_d^{(0)}$, coincide propio con E_d.

L'algoritmo (vedi pseudo-codice 2.2) lavora facendo variare p da 2 a 0 minimizzando di volta in volta la funzione approssimante $E_d^{(p)}$ a partire dal minimo trovato al passo precedente. In particolare la minimizzazione di ogni $E_d^{(p)}$ avviene muovendosi –dal punto di partenza– lungo la direzione del gradiente negativo finchè non si raggiunge un punto stazionario, secondo uno schema noto come NL-SOR (*Non-Linear Successive Over Relaxation*) [bz87].

Si è detto che la prima approssimazione, $E_d^{(2)}$, è convessa, pertanto come punto iniziale al momento dell'avvio può esserne utilizzato uno qualsiasi. Quando $p = 0$ l'approssimante $E_d^{(p)}$, coincide con l'approssimata E_d ed il minimo trovato corrisponde con buona precisione al minimo globale; quindi l'algoritmo termina.

Algoritmo 2.2 GNC

Require: $\vec{\mathbf{y}}$ l'immagine osservata

$\vec{\mathbf{x}} \leftarrow \vec{\mathbf{y}}$

$p = 2$

while $p \neq 0$ **do** $\{E_d^{(p)} \neq E_d = E_d^{(0)}\}$

$\vec{\mathbf{x}} \leftarrow \arg\left(\min E_d^{(p)} \text{ trovato partendo da } \vec{\mathbf{x}}\right)$

$p \leftarrow p - r$ $\{r$ è un fattore di riduzione$\}$

end while

Vediamo ora come sono definite le funzioni approssimanti $E_d^{(p)}$ della (2.7). Innanzi tutto, si può notare che il termine (2.1) nella (2.7) è convesso e la somma di due funzioni convesse è ancora convessa. Pertanto per trovare un'approssimazione convessa della (2.7) è sufficiente trovarne una per la funzione di interazione g data dalla (2.8). Questo consente di costruire le approssimanti, modificando semplicemente la parte relativa alla g.

Consideriamo prima il problema di trovare una serie di funzioni C^1 che approssimino la (2.8). I punti di discontinuità della sua derivata prima sono $t = \frac{\sqrt{\alpha}}{\lambda}$ e $t = -\frac{\sqrt{\alpha}}{\lambda}$. Si può quindi regolarizzarla adattando due archi quadratici in corrispondenza di questi.

Formalmente per $p \in (0, 1]$ si ha che:

$$g^{(p)}(t) = \begin{cases} \lambda^2 t^2 & \text{se } |t| < q(p), \\ \alpha - \dfrac{\tau(p)}{2}\left(|t| - r(p)\right)^2 & \text{se } q(p) \leq |t| \leq r(p), \\ \alpha & \text{se } |t| > r(p), \end{cases} \tag{2.9}$$

dove

$$q(p) = \frac{\sqrt{\alpha}}{\lambda^2}\left(\frac{2}{\tau(p)} + \frac{1}{\lambda^2}\right)^{-\frac{1}{2}}, \quad \tau(p) = \frac{\tau^*}{p}, \quad \tau^* \in \mathbb{R}, \quad \tau^* \in \left(0, \frac{1}{4}\right),$$

e

$$r(p) = \frac{\alpha}{\lambda^2 q(p)}.$$

In Figura 2.3 è riportato il grafico della (2.9) per $p = 1$.

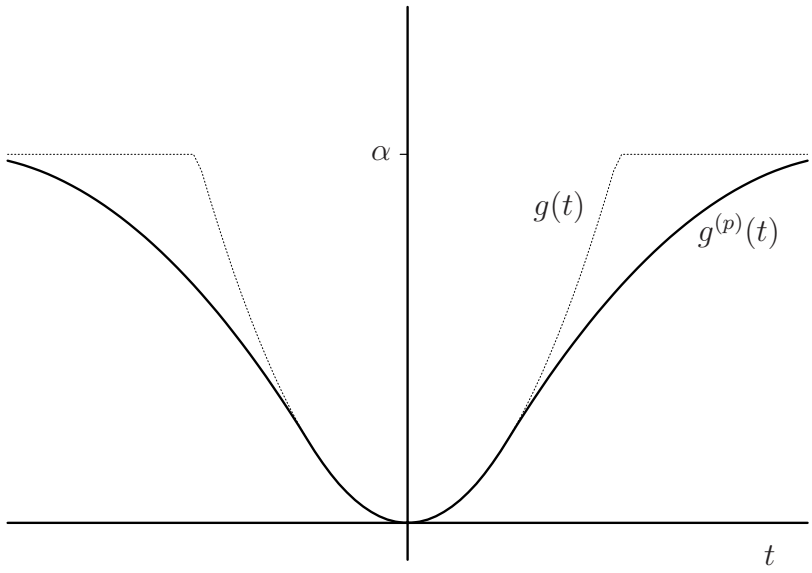


Figura 2.3: Funzione $g^{(p)}(t)$ per $\alpha = 10$ $\lambda = 0.6$ $\tau^* = 0.2$ $p = 1$.

La prima approssimazione convessa di g in C^1 è invece definita, seguendo il modello di M. Nikolova [Nik99], come segue:

$$g^*(t) = \begin{cases} \lambda^2 t^2 & \text{se } |t| < q(1), \\ 2\lambda^2 q(1)|t| - \lambda^2 q(1)^2 & \text{se } |t| \geq q(1). \end{cases} \tag{2.10}$$

Infine per $p \in (1, 2]$ le funzioni approssimanti di g sono date da

$$g^{(p)}(t) = (p-1)g^*(t) + (2-p)g^{(1)}(t), \tag{2.11}$$

con g^* e $g^{(1)}$ definite come sopra (vedi Figura 2.4 per $p = 2$).

Non-linear successive over relaxation

L'algoritmo 2.2 richiede per ogni valore di p la minimizzazione della funzione approssimante $E_d^{(p)}$. Come accennato in precedenza la minimizzazione delle varie approssimazioni avviene iterativamente utilizzando un algoritmo basato sulla discesa del gradiente (NL-SOR). In particolare la k-esima iterata di questo

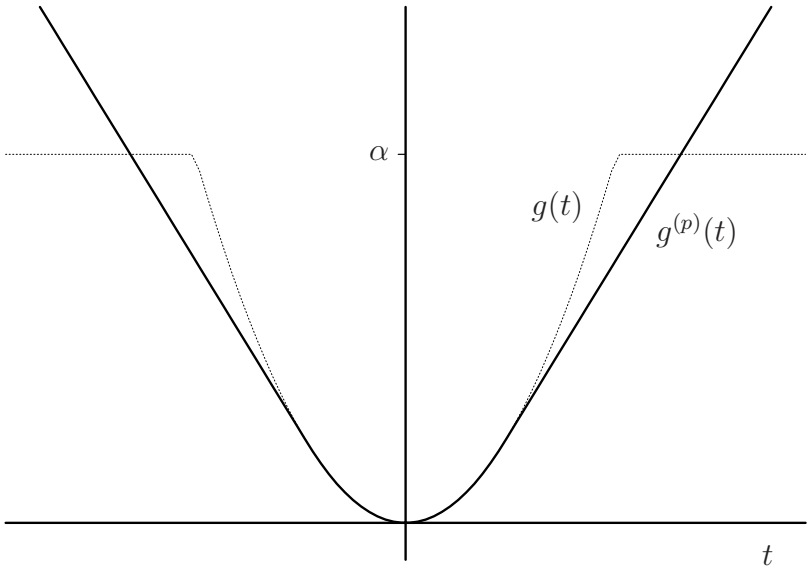


Figura 2.4: Funzione $g^{(p)}(t)$ per $\alpha = 10$ $\lambda = 0.6$ $\tau^* = 0.2$ $p = 2$.

è data per ogni $i = 1, \ldots, n$, da

$$x_i^{(k)} = x_i^{(k-1)} - \frac{\omega}{T_i} \frac{\partial E_d^{(p)}(\vec{\mathbf{x}})}{\partial x_i^{(k-1)}},$$

dove $0 < \omega < 2$ è un parametro che governa la velocità di convergenza e

$$T_i \geq \frac{\partial^2 E_d^{(p)}(\vec{\mathbf{x}})}{\partial x_i^2},$$

è un limite per la derivata seconda. In appendice A è riportato il calcolo delle derivate parziali delle funzioni approssimanti $E_d^{(p)}$.

Il nonlinear SOR è stato dimostrato essere convergente per funzioni strettamente convesse definite in $\mathbb{R}^n$. In particolare si assume che:

i) D sia un dominio convesso in $\mathbb{R}^n$;

ii) $\phi : D \to \mathbb{R}$ sia una funzione strettamente convessa cosicchè, per ogni x, y

in D e per ogni $\lambda \in (0,1)$,

$$\lambda\phi(x) + (1-\lambda)\phi(y) - \phi(\lambda x + (1-\lambda)y) \geq 0,$$

con l'uguaglianza che vale solo quando $x = y$;

iii) ϕ sia di classe C^2 su D.

iv) esista $\gamma \in \mathbb{R}$ tale che $S_\gamma = \{x \in D : \phi(x) \leq \gamma\}$ sia non vuoto e compatto.

v) $F_{ii}(y) \neq 0$ per $i = 1, \ldots, n$ e $y \in S_\gamma$, a meno che y sia il punto in cui ϕ raggiunge il suo minimo (dove $[F_{ij}(y)]$, $(i = 1, \ldots, n; j = 1, \ldots, n)$ denota la matrice Hessiana di ϕ in y).

Dalla ii) e iii) si ricava che la matrice Hessiana di ϕ è semidefinita positiva. Inoltre gli insiemi S_γ sono convessi per ogni $\gamma \in \mathbb{R}$. La iv) implica che ϕ raggiunge il suo minimo in un qualche punto $x^* \in D$. Al contrario, se i), ii) e iii) sono soddisfatte e ϕ raggiunge il suo minimo in un qualche $x^* \in D$ la iv) non è altrettanto banalmente soddisfatta. Per la ii) il punto di minimo di ϕ è unico ed un punto $x^* \in D$ è il punto di minimo di ϕ se e solo se il gradiente $\phi(x^*) = 0$.

Con queste ipotesi è possibile definire il seguente teorema.

Teorema. *Sia* $\{x^k\}$ *la sequenza generata da ogni* $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ *in* S_γ *tale che*

$$x_j^{k+1} = x_j^k, \qquad j \neq i_k$$

e

$$x_{i_k}^{k+1} = x_{i_k}^k - \omega_k \frac{F_{i_k}(x^k)}{F_{i_k i_k}(x^k)}.$$

dove F_i *denota la componente del gradiente di* ϕ *nella direzione della* i*-esima coordinata e* i_k *è uno dei numeri interi* $1, 2, \ldots, n$.

Sia inoltre $\{I_k\}$ *definita da*

$$I_k = \{x : \phi(x) \leq \phi(x^k) \quad e \quad x_j = x_j^k, j \neq i_k\}$$

e sia δ_k *definita da*

$$\delta_k = \frac{F_{i_k i_k}(x^k)}{\max\limits_{y \in I_k} F_{i_k i_k}(y)}.$$

Allora la sequenza $\{x^k\}$ *è ben definita e converge a* x^* *sia se 1) esiste* $\delta > 0$ *tale che* $0 < \delta \leq \omega_k \leq 2\delta_k - \delta$, $k = 0, 1, 2, \ldots$ *sia se 2) esiste* $\psi : \mathbb{R}^n \to \mathbb{R}$ *tale che* ψ *è continua, non negativa,* $\psi(v) = 0$ *se e solo se* $v = 0$ *e*

$$0 \leq \psi(F(x^k)) \leq \omega_k \leq 2\delta_k - \psi(F(x^k)) \leq 2, \quad k = 0, 1, 2, \ldots$$

ammesso che ogni coordinata direzionale sia stata scelta un infinito numero di volte nel precedente processo iterativo.

Dimostrazione. Mostriamo prima che la sequenza $\{x^k\}$ è ben definita e che $\{\phi(x^k)\}$ è convergente. Sia $\bar{x}^0$ il punto finale del segmento di linea I_0 distinto da x^0. Chiaramete x^1 e $\bar{x}^0$ sono dalla stessa parte di x^0. Quindi applicando il teorema di Taylor a ϕ, per qualche $y \in I_0$ si ha:

$$0 = \phi(\bar{x}^0) - \phi(x^0) = F_{i_0}(x^0)(\bar{x}^0_{i_0} - x^0_{i_0}) + \frac{1}{2} F_{i_0 i_0}(y)(\bar{x}^0_{i_0} - x^0_{i_0})^2.$$

Da cui

$$\frac{2|F_{i_0}(x^0)|}{\max\limits_{y \in I_0} F_{i_0 i_0}(y)} \leq |\bar{x}^0_{i_0} - x^0_{i_0}|.$$

Da $\omega_k \leq 2\delta_k$, si ottiene

$$|x^1_{i_0} - x^0_{i_0}| \leq |\bar{x}^0_{i_0} - x^0_{i_0}|.$$

Questo implica che $x^1 \in I_0 \subset S_\gamma$. Per induzione si può mostrare che $\{x^k\}$ è contenuta in S_γ e quindi è ben definita. Inoltre $\{\phi(x^k)\}$ è non crescente e limitata inferiormente, e quindi converge. Mostriamo ora che $\{x^k\}$ converge. Applicando il teorema di Taylor a ϕ,

$$\phi(x^k) - \phi(x^{k+1}) = F_{i_k}(x^k)(x^k_{i_k} - x^{k+1}_{i_k}) - \frac{1}{2} F_{i_k i_k}(y)(x^k_{i_k} - x^{k+1}_{i_k})^2$$

per qualche $y \in (x^k, x^{k+1}) \subseteq I_k$. Sostituendo per x^{k+1} in termini di x^k,

$$\begin{aligned}\phi(x^k) - \phi(x^{k+1}) &= \frac{F_{i_k i_k}(x^k)}{\omega_k}(x^k_{i_k} - x^{k+1}_{i_k})^2 - \frac{1}{2}F_{i_k i_k}(y)(x^k_{i_k} - x^{k+1}_{i_k})^2 \geq \\ &\geq \frac{m}{4}\left(2 - \frac{\omega_k}{\delta_k}\right)(x^k_{i_k} - x^{k+1}_{i_k})^2,\end{aligned}$$

dove $m = \inf\left\{F_{i_k i_k(x^k)}, k = 0, 1, 2, \dots\right\}$. Dal momento che l'Hessiano di ϕ è semidefinito positivo, si ha che $m \geq 0$. Se $m = 0$, allora $x^k \to x^*$, se invece $m > 0$, si possono distinguere due casi.

Caso 1. $\delta \leq \omega_k \leq 2\delta_k - \delta$ per qualche $\delta > 0$. Allora la convergenza di $\left\{\phi(x^k)\right\}$ implica $|x^k_{i_k} - x^{k+1}_{i_k}| \to 0$.

Poichè $\left\{x^k\right\} \in S_\gamma$ (un insieme chiuso e limitato) esiste un punto limite $\bar{x}$ della sequenza $\left\{x^k\right\}$. Definiamo $I = \{i : i \in \{1, 2, \dots, n\}$ tali che $F_i(\bar{x}) = 0\}$ e $J = \{1, 2, \dots, n\} - I$. Definiamo inoltre gli insiemi H_i, $i = 1, 2, \dots, n$ come $H_i = \{u \in S_\gamma : F_i(u) = 0\}$[1]. Sia $J = \{j_1, \dots, j_L\}$, $I = \{i_1, \dots, i_{n-L}\}$ e $H_J = \bigcup_{j \in J} H_j$. Poichè H_J è chiuso esiste $\tau > 0$ tale che:

$$y \in H_J \Rightarrow |\bar{x} - y| \geq \tau.$$

Ricordiamo che $x^k_{i_k} - x^{k+1}_{i_k} \to 0$. Quindi esiste un intero N tale che $k \geq N$ implica

$$|x^k - x^{k+1}| < \frac{\tau}{2}.$$

Sia $B_\delta \subset \mathbb{R}^n$ l'insieme di tutte le x tali che:

$$|(x_{j_1}, \dots, x_{j_L}) - (\bar{x}_{j_1}, \dots, \bar{x}_{j_L})| < \delta \tag{2.12}$$

e

$$\phi(x) < \bar{\phi} + \delta = \phi(\bar{x}) + \delta. \tag{2.13}$$

Osserviamo che per ogni $\epsilon > 0$ possiamo scegliere $\delta(\epsilon)$ in modo tale che

[1]Notare che gli insiemi H_i sono chiusi e non vuoti.

se $x \in B_\delta$ allora $|x - \bar{x}| < \epsilon$. Infatti se questa affermazione non è vera, allora per ogni sequenza $\{\delta_i\}$ decrescente a zero esiste $\{z^1\}$ tale che $z^1 \in B_\delta$ e $|z^1 - \bar{x}| \geq \epsilon$. Tutte le $\{z^1\}$ sono contenute in $S_{\bar{\phi}+\delta_i}$ che è compatto e quindi esiste un punto limite z^0 di $\{z^1\}$ per cui $|z^0 - \bar{x}| \geq \epsilon$. Inoltre per la (2.12), $|(z^0_{j_1}, \ldots, z^0_{j_L}) - (\bar{x}_{j_1}, \ldots, \bar{x}_{j_L})| = 0$ e per la (2.13), $\phi(z^0) = \bar{\phi} = \phi(\bar{x})$. Pertanto

$$\left\langle \text{grad } \phi(\bar{x}), \bar{x} - z^0 \right\rangle = \phi(\bar{x}) - \phi(z^0) = 0$$

che contraddice l'ipotesi ii). Ora poichè

$$x^k_{i_k} - x^{k+1}_{i_k} = \omega_k \frac{F_{i_k}(x^k)}{F_{i_k i_k}(x^k)}$$

segue che $F_{i_k}(x^k) \to 0$ e quindi I non è vuoto.

A questo punto se J è vuoto, $\bar{x} = x^*$ e dal momento che $\left\{\phi(x^k)\right\}$ non è crescente, $x^k \to x^*$. Se invece J non è vuoto, sia $\Lambda = \min_{j \in J}(|F_j(\bar{x})|)$. Per la continuità di F_j esiste un $\epsilon > 0$ tale che $|\bar{x} - y| < \epsilon$ implica $|F_j(y)| > \frac{\Lambda}{2}$. Per le considerazioni fatte sopra esiste un $\epsilon' > 0$ per il quale

$$y \in B_{\epsilon'} \Rightarrow |\bar{x} - y| < \epsilon$$

Infine esiste un intero N_1 per il quale $k \geq N_1$ implica $|F_{i_k}(x^k)| < \frac{\Lambda}{2}$ ed un $p \geq N_1$ tale che $x^p \in B_{\epsilon'}$. Ora $i_p \in I$ e quindi $x^{p+1} \in B_{\epsilon'}$. Per induzione si ha che $x \in B_{\epsilon'}$ per $q \geq p$. Questo impone che $i_q \notin J$ per $q \geq p$ e quindi J è vuoto ma questa è una contraddizione. Pertanto $x_k \to x^*$.

Caso 2. $\psi(F(x^k)) \leq \omega_k \leq 2\delta_k - \psi(F(x^k))$. In questo caso o $F(x^k)$ è limitato lontanto da zero e quindi si ritorna nel caso 1, oppure una sottosequenza di $F(x^k)$ converge a zero il che implica $x^k \to x^*$. □

Complessità

Il costo computazionale dell'algoritmo 2.2 è k volte il costo dell'algoritmo NL-SOR, con k uguale al numero di approssimazioni da minimizzare. La complessità del NL-SOR può essere stimata come $O(\lambda N^2)$, dove N^2 è il numero di

pixel dell'immagine e λ è il parametro di regolarizzazione definito nella (2.2). Quindi la complessità dell'algoritmo 2.2 è $O(k\lambda N^2)$.

2.3 Restauro cieco

Nei paragrafi precedenti si è visto che, quando si conosce l'operatore di sfocatura A, il problema del restauro d'immagine può essere riformulato come un problema di minimizzazione di una funzione non globalmente convessa. In particolare il risultato è dato da quell'immagine $\vec{\mathbf{x}}^*$ tale che

$$\vec{\mathbf{x}}^* = \arg\min_{\vec{\mathbf{x}}} E_d(A, \vec{\mathbf{x}}).$$

con $E_d(A, \vec{\mathbf{x}})$ definita come in (2.7) e A noto.

Molto spesso tuttavia le immagini con cui si lavora presentano difetti dovuti a processi di degradazione sconosciuti o difficilmente quantificabili. In questi casi l'operatore di sfocatura è chiaramente non noto e il problema che cerca di trovarne una stima è conosciuto come *restauro cieco*.

Formalmente l'obiettivo del restauro cieco consiste nel trovare quella coppia $(A^*, \vec{\mathbf{x}}^*)$ tale che

$$(A^*, \vec{\mathbf{x}}^*) = \arg\min_{(A,\vec{\mathbf{x}})} E_d(A, \vec{\mathbf{x}}). \tag{2.14}$$

La minimizzazione congiunta della funzione energia non è banale poichè questa è fortemente non lineare. In letteratura esistono comunque alcuni metodi [Ton01].

Una tecnica alternativa per risolvere la (2.14) è quella di minimizzare in successione l'energia duale rispetto $\vec{\mathbf{x}}$ e A come mostrato nell'algoritmo 2.3.

Osservazione. Per valori di λ piccoli l'algoritmo ha un punto fisso banale, vicino a $(I, \vec{\mathbf{y}})$ dove I è la matrice identità appartenente a $\mathbb{R}^{(2h+1)\times(2h+1)}$, $h \in \mathbb{N}^+$. Per evitare questo problema è necessario partire da un punto che sia vicino alla soluzione e pertanto avere una buona stima dell'operatore di blur originale.

In questa tesi si è utilizzato un approccio differente [GMMM08] basato sull'osservazione che la soluzione della (2.14) coincide con quella del seguente

Algoritmo 2.3 Restauro Cieco

Require: A una stima dell'operatore di sfocatura
Require: $\vec{\mathbf{y}}$ l'immagine osservata
  $\vec{\mathbf{x}} \leftarrow \vec{\mathbf{y}}$
  while non è stato trovato un punto fisso **do**
    $\vec{\mathbf{x}} \leftarrow \arg\min_{\vec{\mathbf{x}}} E_d(A, \vec{\mathbf{x}})$
    $A \leftarrow \arg\min_{A} E_d(A, \vec{\mathbf{x}})$
  end while

problema di minimizzazione annidato:

$$A^* = \arg\min_{A} E_d(A, \vec{\mathbf{x}}(A)), \tag{2.15}$$

dove

$$\vec{\mathbf{x}}(A) = \arg\min_{\vec{\mathbf{x}}} E_d(A, \vec{\mathbf{x}}), \tag{2.16}$$

e

$$\vec{\mathbf{x}}^* = \vec{\mathbf{x}}(A^*).$$

Ora calcolare la (2.16) corrisponde a risolvere il problema inverso associato all'operatore di sfocatura A. Per trovarne una soluzione si è utilizzato l'algoritmo 2.2. Per risolvere la (2.15), si è invece utilizzata una tecnica conosciuta come PSO (*Particle swarm optimization*).

Prima di vedere in dettaglio come è stato implementato l'algoritmo forniamo una panoramica del PSO.

3

Ottimizzazione a sciame di particelle

Il particle swarm optimization è una tecnica di ottimizzazione stocastica, basata su una popolazione di individui, inventata da Russel Eberhart e James Kennedy nel 1995 [KE95]. Fa parte della famiglia degli algoritmi evolutivi e può essere utilizzato per risolvere diversi problemi che vanno dall'allenamento di reti neurali alla minimizzazione di funzioni non convesse. La versione originale fu inspirata dal comportamento sociale di stormi di uccelli in movimento, con l'obiettivo di trovare il modello che permette a questi di volare in sincrono e cambiare improvvisamente direzione per poi raggrupparsi in una nuova formazione ottimale. In particolare, l'idea che sta alla base del PSO è che ogni individuo in un gruppo, di fronte ad un particolare problema, tende ad interagire con gli altri per risolverlo e man mano che le interazioni si susseguono, si modificano le credenze, le attitudini e i comportamenti di ciascuno di questi.

Nel PSO, gli individui, a cui ci si riferisce con il termine particelle, vengono letteralmente fatti "volare" in uno spazio di ricerca multidimensionale (es. $\mathbb{R}^n$). I movimenti delle particelle all'interno di questo sono influenzati dalla tendenza socio-psicologica che porta ogni individuo ad emulare il successo degli altri. La ricerca di una singola particella è pertanto collegata a quella delle altre particelle all'interno del gruppo.

3.1 Caratteristiche fondamentali

Il PSO cerca di simulare la cooperazione sociale appena descritta. È dato un problema ed esiste un modo per valutare l'ottimalità di una data soluzione nella forma di una funzione obiettivo che chiamiamo $f : \mathbb{R}^n \to \mathbb{R}$ (*funzione di fitness*). È inoltre definita una struttura di comunicazione che assegna ad ogni individuo un insieme di vicini con cui interagire. L'algoritmo mantiene poi una popolazione –detta anche sciame– di particelle ognuna delle quali ha una posizione che corrisponde ad una possibile soluzione del problema. Ogni particella è in grado di valutare, tramite la funzione obiettivo, la bontà della propia posizione e di ricordare la migliore visitata. Questa informazione è condivisa tra tutti i vicini cosicchè ogni particella conosce anche la migliore posizione di tutti i vicini con cui interagisce. Gli individui all'interno dello sciame hanno un comportamento molto semplice: modificano la propia posizione in funzione della loro esperienza e di quella dei loro vicini.

Sia $\vec{\mathbf{x}}_i^{(t)}$ la posizione della particella i nello spazio di ricerca al tempo t, con t discreto. La posizione di i è aggiornata aggiungendo una velocità, $\vec{\mathbf{v}}_i^{(t)}$, alla posizione corrente, ovvero

$$\vec{\mathbf{x}}_i^{(t+1)} = \vec{\mathbf{x}}_i^{(t)} + \vec{\mathbf{v}}_i^{(t+1)}, \tag{3.1}$$

con $\vec{\mathbf{x}}_i^{(0)} \sim U(x_{min}, x_{max})$.

È quindi il vettore velocità quello che guida il processo di ottimizzazione, e riflette sia l'esperienza personale sia quella acquisita dalle interazioni sociali. Questi due fattori vengono rispettivamente chiamati *componente cognitiva* e *componente sociale*.

Esistono due versioni di PSO chiamate *gbest* e *lbest* PSO [eds96]. La differenza tra le due è data dall'insieme dei vicini con cui una data particella interagisce direttamente.

3.1.1 Global best PSO

Nel gbest PSO ogni particella scambia informazioni con tutte le altre. La struttura di comunicazione è definita dalla tipologia a stella (vedi Figura 3.4(a)) e

la componente sociale nel calcolo della velocità riflette l'informazione ottenuta da tutte le particelle.

Algoritmo 3.1 *gbest* PSO

Require: creazione ed inizializzazione di uno sciame n-dimensionale
  repeat
    for all $i \in s$ **do**
      if $f(\vec{\mathbf{x}}_i) < f(\vec{\mathbf{y}}_i)$ **then**
        $\vec{\mathbf{y}}_i \leftarrow \vec{\mathbf{x}}_i$
      end if
      if $f(\vec{\mathbf{y}}_i) < f(\hat{\mathbf{y}})$ **then**
        $\hat{\mathbf{y}} \leftarrow \vec{\mathbf{y}}_i$
      end if
    end for
    for all $i \in s$ **do**
      aggiorna la velocità di i con la (3.2)
      aggiorna la posizione di i con la (3.1)
    end for
  until non è verificata una condizione di uscita

Se si indica con $\hat{\mathbf{y}}^{(t)}$ la miglior posizione trovata dallo sciame, il vettore velocità associato alla particella i può essere calcolato per ogni dimensione $j = 1, \ldots, n$ come:

$$v_{i,j}^{(t+1)} = v_{i,j}^{(t)} + c_1 r_{1,j}^{(t)} \left[y_{i,j}^{(t)} - x_{i,j}^{(t)} \right] + c_2 r_{2,j}^{(t)} \left[\hat{y}_j^{(t)} - x_{i,j}^{(t)} \right], \tag{3.2}$$

dove $v_{i,j}^{(t)}$ è la j-esima componente del vettore $\vec{\mathbf{v}}_i$ al tempo t, $x_{i,j}^{(t)}$ è la posizione della particella i nella dimensione j al tempo t, c_1 e c_2 sono costanti positive, chiamate *coefficienti di accelerazione*, utilizzate per scalare rispettivamente il contributo della componente cognitiva e sociale, e $r_{1,j}^{(t)}, r_{2,j}^{(2)} \sim U(0,1)$ sono sequenze casuali che definiscono la natura stocastica dell'algoritmo.

La miglior posizione personale di ogni particella i, $\vec{\mathbf{y}}_i$ è quella che detiene il maggior grado di ottimalità tra tutte quelle visitate da i. Se si considera un problema di minimizzazione questa può essere definita come:

$$\vec{\mathbf{y}}_i^{(t+1)} = \begin{cases} \vec{\mathbf{y}}_i^{(t)} & \text{se } f(\vec{\mathbf{x}}_i^{(t+1)}) \geq f(\vec{\mathbf{y}}_i^{(t)}), \\ \vec{\mathbf{x}}_i^{(t+1)} & \text{se } f(\vec{\mathbf{x}}_i^{(t+1)}) < f(\vec{\mathbf{y}}_i^{(t)}). \end{cases}$$

La miglior posizione dello sciame è invece data da:

$$\hat{\mathbf{y}}^{(t)} = \arg \min_{\vec{\mathbf{x}} \in \{\vec{\mathbf{y}}_0, \ldots, \vec{\mathbf{y}}_s\}} f(\vec{\mathbf{x}}),$$

con $s \in \mathbb{N}$ uguale al numero di particelle nella popolazione. Lo pseudo-codice del gbest PSO è riportato nell'algoritmo 3.1.

3.1.2 Local best PSO

Il local best PSO utilizza la struttura di comunicazione ad anello (vedi Figura 3.4(b)) dove ogni particella comunica con un sottoinsieme di vicini. La componente sociale riflette l'informazione scambiata tra questi, evidenziando la natura locale del sistema.

La velocità è calcolata come:

$$v_{i,j}^{(t+1)} = v_{i,j}^{(t)} + c_1 r_{1,j}^{(t)} \left[y_{i,j}^{(t)} - x_{i,j}^{(t)} \right] + c_2 r_{2,j}^{(t)} \left[\hat{y}_{i,j}^{(t)} - x_{i,j}^{(t)} \right], \tag{3.3}$$

dove $\hat{y}_{i,j}$ è la migliore posizione trovata (nella dimensione j) dal sottoinsieme di particelle che comunica con i. Indicando questa con $\hat{\mathbf{y}}_i$ si ha che:

$$\hat{\mathbf{y}}_i^{(t)} = \arg \min_{\vec{\mathbf{x}} \in \mathcal{N}_i} f(\vec{\mathbf{x}}),$$

con $\mathcal{N}_i$ definito come:

$$\mathcal{N}_i = \left\{ \vec{\mathbf{y}}_{i-s}^{(t)}, \vec{\mathbf{y}}_{i-s+1}^{(t)}, \ldots, \vec{\mathbf{y}}_{i-1}^{(t)}, \vec{\mathbf{y}}_i^{(t)}, \vec{\mathbf{y}}_{i+1}^{(t)}, \ldots, \vec{\mathbf{y}}_{i+s}^{(t)} \right\}$$

e $s \in \mathbb{N}$ uguale alla cardinalità del sottoinsieme.

Nonostante esistano strategie basate sulla similarità spaziale, i vicini di ogni particella vengono tipicamente scelti in base ai loro indici. Le ragioni che spingono a far questo sono essenzialmente due:

1. Si riduce la complessità computazionale, poichè non è necessario calcolare la distanza Euclidea tra tutte le coppie di particelle.

2. Si favorisce la diffusione dell'informazione a tutte le particelle, indipendentemente dalla loro posizione nello spazio di ricerca.

Bisogna anche notare che i sottoinsiemi di comunicazione si sovrappongono. Una particella può essere membro di diversi sottoinsiemi favorendo l'interconnessione e lo scambio d'informazione tra questi. Questo consente allo sciame di convergere verso un unico punto che corrisponde all'ottimo globale del problema. Il global best PSO può essere visto come un caso speciale del local best PSO in cui s è uguale alla cardinalità dello sciame.

L'algoritmo 3.2 riassume il local best PSO.

Algoritmo 3.2 *lbest* PSO

Require: creazione ed inizializzazione di uno sciame n-dimensionale

```
repeat
    for all i ∈ s do
        if f(x⃗_i) < f(y⃗_i) then
            y⃗_i ← x⃗_i
        end if
        if f(y⃗_i) < f(ŷ_i) then
            ŷ_i ← y⃗_i
        end if
    end for
    for all i ∈ s do
        aggiorna la velocità di i con la (3.3)
        aggiorna la posizione di i con la (3.1)
    end for
until non è verificata una condizione di uscita
```

3.1.3 Confronto

Le due versioni di PSO appena discusse sono simili nel senso che entrambe convergono verso la migliore posizione trovata dallo sciame. Le differenze tra i due approcci sono essenzialmente due e riguardano la loro convergenza [eds96, KVR02]:

- Grazie all'interconnessione tra tutte le particelle il global best PSO converge più velocemente. Tuttavia questa maggiore velocità di convergenza si paga in termini di una minore esplorazione.

- Come conseguenza della sua maggiore diversità (che si manifesta nella copertura di più zone nello spazio di ricerca), il local best PSO è meno soggetto a rimanere intrappolato in un minimo locale.

3.1.4 Analisi della velocità

Il calcolo della velocità dato dalle equazioni (3.2) e (3.3) consiste nella somma di tre termini:

- La velocità precedente, $\vec{\mathbf{v}}_i^{(t)}$, che serve alla particella per ricordare la direzione del suo movimento. In particolare questo termine può essere visto come un momento che impedisce di cambiare drasticamente direzione. Questa componente è anche chiamata inerzia.

- La componente cognitiva, $c_1\vec{\mathbf{r}}_1(\vec{\mathbf{y}}_i - \vec{\mathbf{x}}_i)$, che può essere vista come il ricordo della miglior posizione raggiunta. L'effetto di questo termine è l'attrazione di ogni particella i verso l'ottimo trovato da ciascuna, in accordo alla tendenza degli individui a tornare nei luoghi e nelle situazioni che più li hanno soddisfatti. Kennedy e Eberhart si riferiscono alla componente cognitiva anche con il termine "nostalgia" [KE95].

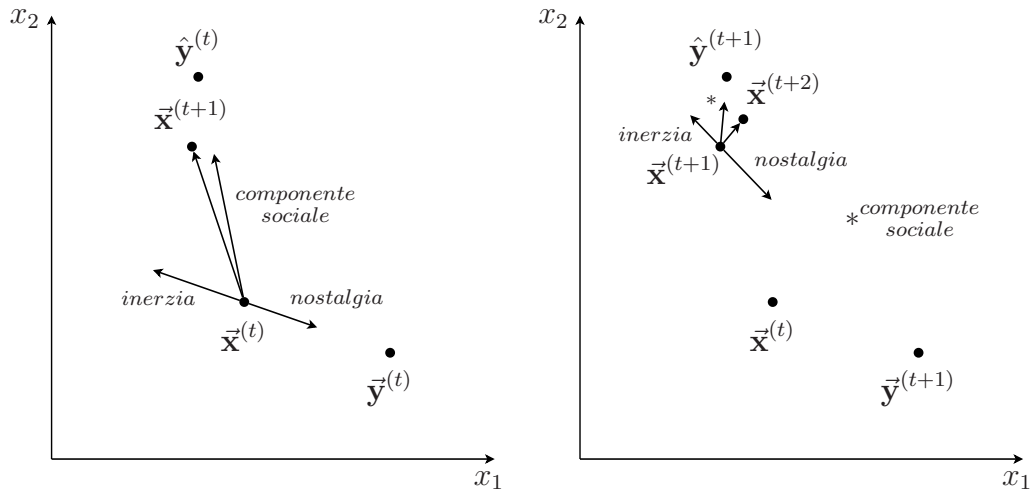
- La componente sociale, $c_2\vec{\mathbf{r}}_2(\hat{\mathbf{y}} - \vec{\mathbf{x}}_i)$, nel caso del global best PSO o, $c_2\vec{\mathbf{r}}_2(\hat{\mathbf{y}}_i - \vec{\mathbf{x}}_i)$, nel caso del local best PSO, che quantifica il successo della particella i in relazione all'intero gruppo o all'insieme dei vicini con cui comunica. Concettualmente la componente sociale rappresenta una norma o uno standard a cui gli individui tendono ad attenersi. L'effetto di questo termine è l'attrazione delle particelle verso la miglior posizione trovata dall'intero gruppo o dal gruppo di vicini.

3.1.5 Rappresentazione geometrica

L'effetto del vettore velocità può essere facilmente illustrato se si considera una singola particella che si move in uno spazio di ricerca bidimensionale.

Un esempio è mostrato in Figura 3.1(a), dove è rappresentato lo stato dello sciame al tempo t. Da notare come la nuova posizione, $\vec{\mathbf{x}}^{(t+1)}$, sia più

vicina all'ottimo globale $\hat{\mathbf{y}}^{(t)}$. Assumendo che la miglior posizione trovata dalla particella non cambi, al passo $t+1$ la situazione si evolve come mostrato in Figura 3.1(b). Anche in questo caso è da notare come le tre componenti della velocità influiscano sul movimento della particella verso $\hat{\mathbf{y}}^{(t+1)}$.



(a) Passo t (b) Passo $t+1$

Figura 3.1: Rappresentazione geometrica dell'aggiornamento della velocità e della posizione di una singola particella in uno spazio di ricerca bidimensionale.

Può succedere che la particella oltrepassi l'ottimo globale (principalmente a causa dell'inerzia), dando origine a due possibili scenari:

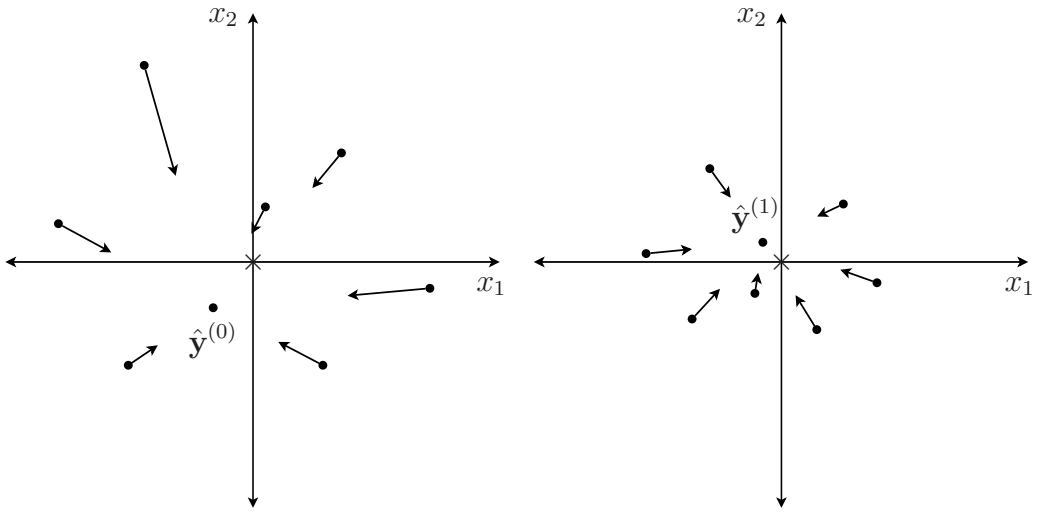
1. La nuova posizione è migliore dell'ottimo globale; ed in questo caso, verrà riconosciuta come la migliore raggiunta dallo sciame.

2. La nuova posizione è peggiore dell'ottimo globale. In questo caso la componente sociale e cognitiva permetteranno alla particella –in successivi passi temporali– di cambiare direzione e di tornare indietro.

L'effetto cumulativo di tutti gli aggiornamenti di posizione è che ogni particella converge ad un punto sulla linea che connette la migliore posizione trovata dallo sciame (o dal gruppo di vicini) e quella trovata dalla particella stessa [vdBE06].

Consideriamo ora il caso con più di una particella concentrando l'attenzione sul global best PSO. Una rappresentazione geometrica di questo è data in

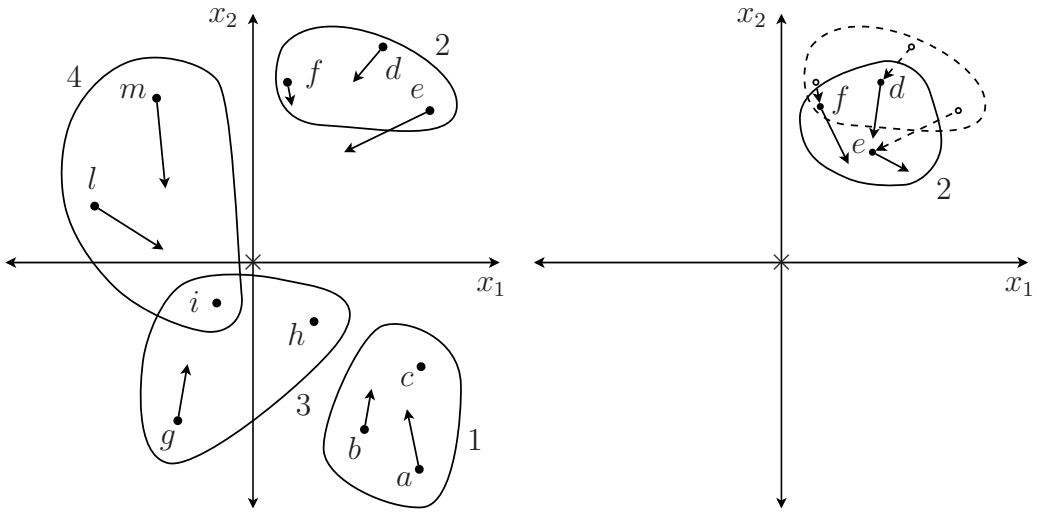
Figura 3.2 in riferimento ad un problema di minimizzazione di una funzione di due variabili x_1 e x_2. La Figura 3.2(a) mostra le posizioni iniziali di otto particelle nello spazio di ricerca. L'ottimo del problema è indicato dal simbolo '$\times$', mentre $\hat{\mathbf{y}}$ specifica ancora una volta la miglior posizione dello sciame. Si assume che $\vec{\mathbf{v}}_i^{(0)} = 0 \quad \forall i$, cossicchè al tempo $t = 0$ solo la componente sociale influisce sull'aggiornamento della posizione di ogni particella i (il contributo della componente cognitiva al passo $t = 0$ è infatti zero). Da notare come la particella associata a $\hat{\mathbf{y}}$ non cambi posizione. In Figura 3.2(b) è possibile vedere le nuove posizioni di tutte le particelle al passo temporale $t = 1$. È stato trovato un nuovo $\hat{\mathbf{y}}$ ed ora tutte le componenti influiscono sul vettore velocità, con le particelle che si muovono verso $\hat{\mathbf{y}}^{(1)}$.



(a) $t = 0$ (b) $t = 1$

Figura 3.2: Rappresentazione geometrica del *gbest* PSO.

Consideriamo infine il local best PSO. La Figura 3.3 mostra come le particelle sono influenzate dai loro vicini immediati. Sono rappresentati solo alcuni dei movimenti ed è indicata solo la direzione della velocità aggregata. Nell'insieme 1 le particelle a e b si muovono verso c. Nel gruppo 2, d e e si muovono verso f. Al passo successivo e sarà la miglior soluzione per il gruppo 2 e d ed f si muoveranno verso questa come mostrato in Figura 3.3(b). Da notare il movimento generale dell'intero gruppo verso il minimo.



(a) Lo sciame iniziale (b) Un secondo sciame

Figura 3.3: Rappresentazione geometrica del *lbest* PSO.

3.1.6 Aspetti dell'algoritmo

Alcuni aspetti degli algoritmi 3.1 e 3.2 non sono ancora stati discussi. Questi includono l'inizializzazione, e la condizione di uscita.

Il primo passo del PSO consiste nel creare lo sciame, inizializzando le particelle e i parametri di controllo. I coefficienti di accelerazione c_1 e c_2, le velocità iniziali, le posizioni iniziali e le migliori posizioni iniziali di ciascuna particella, sono tutti parametri che devono essere specificati[1]. Generalmente le posizioni delle particelle vengono inizializzate in modo tale da coprire uniformemente lo spazio di ricerca. È importante notare che l'efficienza del PSO è influenzata dalla diversità dello sciame iniziale, ovvero da quanto bene sono distribuite le particelle nello spazio. Difficilmente si troverà l'ottimo di un problema se questo cade in una regione non inizialmente coperta.

Assumiamo che l'ottimo debba essere trovato all'interno del dominio definito da due vettori $\vec{\mathbf{x}}_{min}$ e $\vec{\mathbf{x}}_{max}$ che rappresentano rispettivamente il minimo e il massimo valore in ogni dimensione. Un buon metodo per inizializzare la

[1]Nel local best PSO si deve anche specificare la dimensione del gruppo.

posizione delle particelle è il seguente:

$$x_{i,j} = x_{min,j} + r_j \left(x_{max,j} - x_{min,j}\right), \quad \forall j = 1, \ldots, n, \quad \forall i = 1, \ldots, s$$

dove $x_{i,j}$ indica la j-esima componente del vettore posizione ($\in \mathbb{R}^n$) associato alla particella i, $s \in \mathbb{N}$ è la dimensione dello sciame e $r_j \sim U(0,1)$.

Le velocità possono essere inizializzate a zero, ovvero

$$\vec{\mathbf{v}}_i^{(0)} = 0.$$

È anche possibile inizializzare le velocità con valori casuali, ma non è necessario, e quando fatto va gestito con cura. Infatti velocità con valori troppo grandi potrebbero causare movimenti iniziali troppo consistenti che potrebbero indurre lo sciame ad uscire dai limiti dello spazio di ricerca.

La migliore posizione trovata da ciascuna particella è inizializzata alla posizione iniziale, ovvero

$$\vec{\mathbf{y}}_i^{(0)} = \vec{\mathbf{x}}_i^{(0)}.$$

L'altro aspetto non considerato riguarda la condizione di uscita e quindi il criterio utilizzato per terminare il processo di ricerca. Per scegliere questo vanno considerati due fattori:

1. La condizione di uscita non dovrebbe causare una convergenza prematura, o si otterrebbero soluzioni subottime.

2. La condizione di uscita dovrebbe evitare valutazioni inutili della funzione obiettivo. Se il criterio di arresto richiede frequenti valutazioni della funzione di fitness, la complessità del processo di ricerca potrebbe aumentare notevolmente.

Le condizioni di uscita più comuni sono le seguenti:

- Termina dopo un numero massimo di passi o di valutazioni della funzione obiettivo. È chiaro che se il numero massimo di iterazioni è troppo piccolo si potrebbe uscire prima di aver trovato una buona soluzione. Questo

criterio è generalmente usato insieme a un test di convergenza per forzare l'uscita se l'algoritmo non converge entro un limite di passi. Usato da solo questo metodo è utile in quei casi in cui l'obiettivo è valutare la migliore soluzione trovata in un certo periodo di tempo.

- Termina quando una soluzione accettabile è stata trovata. Questo criterio termina il processo di ricerca non appena viene trovata una posizione $\vec{\mathbf{x}}_i$ tale che $f(\vec{\mathbf{x}}_i) \leq |f(\vec{\mathbf{x}}^*) + \epsilon|$, dove $\vec{\mathbf{x}}^*$ è l'ottimo della funzione obiettivo f ed ϵ è l'errore consentito. Il valore di soglia, ϵ, deve essere scelto con cura. Se ϵ è troppo grande, il processo potrebbe terminare in una soluzione subottima. Se invece ϵ è troppo piccolo la ricerca potrebbe non terminare mai. Inoltre questo metodo richiede una conoscenza a priori di quale sia l'ottimo e questo è possibile solo in alcuni problemi.

- Termina quando non ci sono miglioramenti dopo un numero fissato di passi. Ci sono vari modi per valutare un miglioramento. Per esempio se il cambiamento medio delle posizioni è piccolo, si può assumere che lo sciame sia arrivato a convergenza. Oppure se la media delle velocità è prossima allo zero, verranno fatti solo piccoli passi, e quindi la ricerca può essere terminata. Comunque questi metodi introducono altri due parametri per cui bisogna trovare un valore corretto: 1) il numero di iterazioni entro cui valutare se ci sono miglioramenti, 2) una soglia che definisce quando non ci sono progressi da un passo ad un altro.

3.1.7 Strutture di comunicazione

La chiave che guida il PSO è l'interazione sociale. Le particelle nello sciame comunicano le une con le altre e, sulla base della conoscenza acquisita, si muovono per diventare più simili alle loro compagne "migliori". Questo comportamento è tipico del mondo animale, dove un organismo è influenzato dai simili che lo circondano e dove gli individui meno promettenti seguono il modello di quelli con più "successo".

La convergenza del PSO dipende fortemente dalla struttura di comunicazione utilizzata. Con una rete sociale fortemente connessa gran parte degli

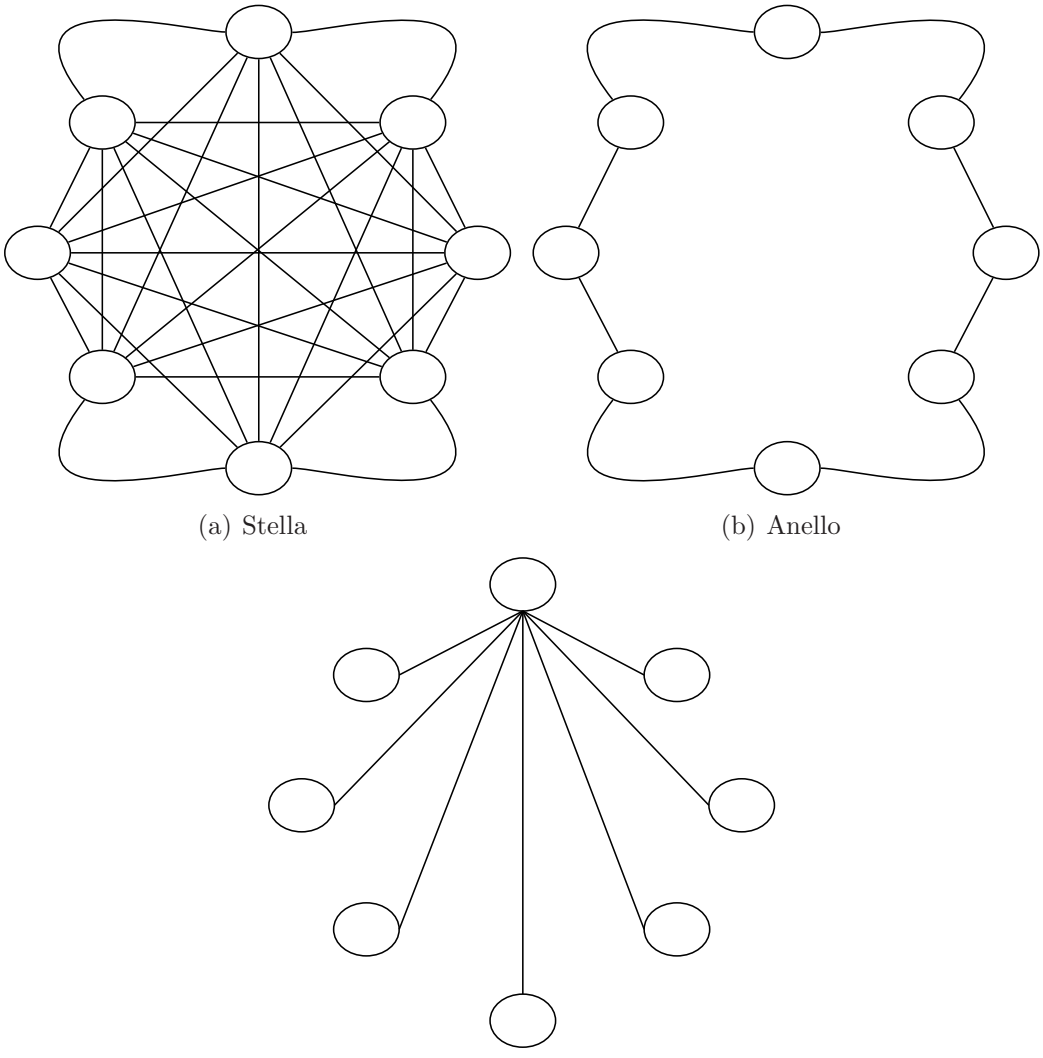
individui sono in grado comunicare gli uni con gli altri con la conseguenza che l'informazione filtra rapidamente all'interno della rete. In termini di ottimizzazione questo si traduce in una maggiore velocità di convergenza. Pertanto, come accennato in precedenza, un PSO che utilizza una struttura di comunicazione fortemente connessa converge rapidamente, ma è più suscettibile a cadere in minimi locali. L'utilizzo di una rete meno connessa consente invece di esplorare più accuratamente lo spazio di ricerca.

Le strutture di comunicazione più comuni in letteratura sono le seguenti:

- La struttura sociale a stella, dove tutte le particelle sono interconnesse come mostrato in Figura 3.4(a). In questo caso ogni particella può comunicare con ogni altra all'interno dello sciame ed è attratta verso la miglior posizione trovata da tutte. La prima versione del PSO usava una rete di questo tipo.

- La struttura sociale ad anello, dove ogni particella può comunicare con i suoi n vicini immediati. Se $n = 2$ ogni particella interagisce con quelle adiacenti come mostrato in Figura 3.4(b). In questo caso le particelle sono attratte e si muovono nella direzione della "migliore" del gruppo. È importante notare che i gruppi di vicini possono sovrapporsi facilitando lo scambio di informazione tra questi e garantendo, alla fine, la convergenza ad una singola soluzione. La struttura ad anello consente di trovare, grazie alla sua maggiore esplorazione, una soluzione qualitativamente migliore rispetto a quella a stella.

- La struttura sociale a ruota, dove gli individui in un gruppo sono isolati l'uno dall'altro. In questo caso una particella funge da punto focale e tutta l'informazione viene scambiata attraverso questa (vedi Figura 3.4(c)). La particella "focale" confronta le posizioni di tutte le particelle nel gruppo e si muove verso la migliore di queste. Se la nuova posizione ha un grado di ottimalità maggiore della precedente, il successo viene comunicato a tutti i membri del gruppo. La struttura a ruota rallenta la propagazione delle soluzioni migliori nello sciame.



(a) Stella

(b) Anello

(c) Ruota

Figura 3.4: Alcuni esempi di strutture di comunicazione.

3.2 Variazioni di base

Esistono in letteratura una serie di modifiche, applicabili alla versione originale del PSO, che permettono di migliorarne la convergenza e la qualità della soluzione trovata. Queste modifiche includono l'introduzione di un controllo della velocità, un fattore d'inerzia, e un coefficiente di costrizione.

3.2.1 Controllo della velocità

Un importante aspetto che determina l'efficienza e l'accuratezza di un algoritmo di ottimizzazione è il compromesso tra la capacità di esplorare zone diverse nello spazio di ricerca e l'abilità di concentrarsi in quelle più promettenti per migliorare una possibile soluzione. Nel PSO questi obiettivi contraddittori sono bilanciati dall'equazione di aggiornamento della velocità.

I tre termini nelle equazioni (3.2) e (3.3) contribuiscono tutti a quantificare la dimensione del passo fatto dalle particelle in ogni istante temporale. Se il passo è troppo grande, può succedere che le particelle abbandonino lo spazio di ricerca e divergano. Per evitare che ciò accada, si fa in modo di limitare le velocità ad un valore prefissato. Se indichiamo con $V_{max,j}$ tale valore nella dimensione j l'equazione di aggiornamento della velocità della particella i può essere riscritta come:

$$v_{i,j}^{(t+1)} = \begin{cases} v_{i,j}^{'(t+1)} & \text{se } v_{i,j}^{'(t+1)} < V_{max,j}, \\ V_{max,j} & \text{se } v_{i,j}^{'(t+1)} \geq V_{max,j}, \end{cases}$$

dove $v_{i,j}^{'}$ è dato dalla (3.2) o (3.3).

Il valore $V_{max,j}$ è molto importante dal momento che controlla la granularità della ricerca. Se è troppo piccolo, lo sciame potrebbe non esplorare a sufficienza lo spazio e rimanere bloccato in un ottimo locale. Se è troppo grande le particelle potrebbero mancare una buona soluzione, continuando a saltare da una zona all'altra. Generalmente i valori di $V_{max,j}$ sono scelti in modo tale da essere una frazione del dominio di ogni dimensione dello spazio di ricerca.

Ovvero

$$V_{max,j} = \delta(x_{max,j} - x_{min,j}),$$

dove $x_{max,j}$ e $x_{min,j}$ sono rispettivamente il massimo e il minimo valore del dominio di $\vec{\mathbf{x}}$ nella dimensione j e $\delta \in (0, 1]$.

Il controllo della velocità non influenza la posizione della particella ma solo il passo e la direzione del movimento. Questo effetto è mostrato in Figura 3.5 considerando una particella i in uno spazio bidimensionale. La posizione $\vec{\mathbf{x}}_i^{(t+1)}$ è il risultato del movimento di i senza il controllo della velocità. Il simbolo $\vec{\mathbf{x}}_i^{'(t+1)}$ denota invece la posizione di i dopo aver limitato la velocità nella seconda dimensione. Da notare come la direzione ed il passo siano cambiati.

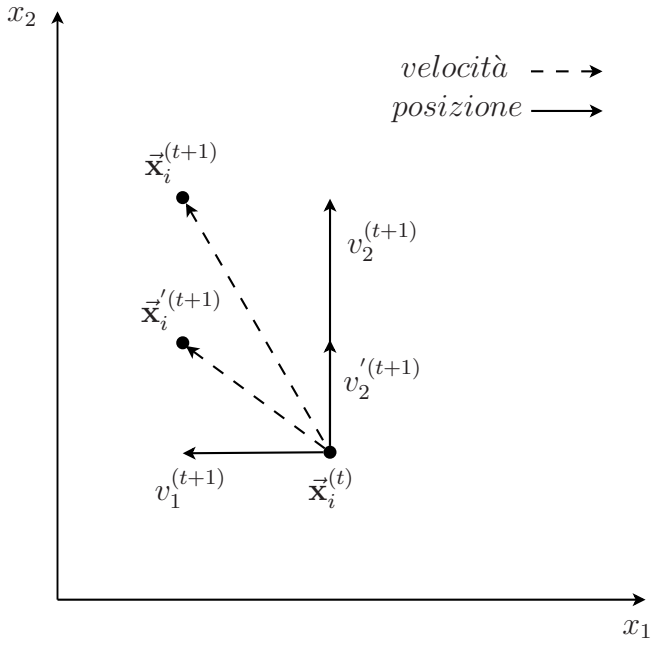


Figura 3.5: Effetto del controllo della velocità.

Un problema connesso al metodo appena descritto si presenta quando tutte le velocità sono uguali al limite massimo prefissato. In questa situazione le particelle rimangono confinate ai bordi di un ipercubo definito da $[\vec{\mathbf{x}}_i^{(t)} - \vec{\mathbf{V}}_{max}, \vec{\mathbf{x}}_i^{(t)} + \vec{\mathbf{V}}_{max}]$ con difficoltà da parte dello sciame a concentrarsi

in quest'area. Per risolvere il problema si può utilizzare un fattore d'inerzia oppure ridurre periodicamente i valori V_{max}, j. L'idea è quella di partire con valori sufficientemente grandi e poi diminuirli man mano che il processo di ottimizzazione avanza. Gli approcci sono essenzialmente due:

1. Cambiare la velocità massima quando la miglior posizione trovata dallo sciame non cambia per τ passi consecutivi:

$$V_{max,j}^{(t+1)} = \begin{cases} \gamma V_{max,j}^{(t)} & \text{se } f\left(\hat{\mathbf{y}}^{(t)}\right) \geq f\left(\hat{\mathbf{y}}^{(t-t')}\right) \quad \forall t' = 1, \ldots, \tau, \\ V_{max,j}^{(t)} & \text{altrimenti,} \end{cases}$$

   dove γ varia da 1 a 0.01.

2. Decrementare esponenzialmente la velocità massima:

$$V_{max,j}^{(t+1)} = \left(1 - \left(\frac{t}{t_{max}}\right)^{\alpha}\right) V_{max,j}^{(t)},$$

   dove α è una costante positiva e t_{max} è il massimo numero di iterazioni.

3.2.2 Fattore d'inerzia

Il fattore d'inerzia fu introdotto da Shi ed Eberhart [SE98] come uno strumento per controllare la capacità di esplorazione dello sciame. In particolare questo (w) scala il contributo dato dalla velocità precedente, controllando quanto la memoria della direzione passata influisce sulla nuova velocità. Per il global best PSO, l'equazione di aggiornamento della velocità cambia dalla (3.2) in

$$v_{i,j}^{(t+1)} = w v_{i,j}^{(t)} + c_1 r_{1,j}^{(t)} \left[y_{i,j}^{(t)} - x_{i,j}^{(t)}\right] + c_2 r_{2,j}^{(t)} \left[\hat{y}_j^{(t)} - x_{i,j}^{(t)}\right]. \tag{3.4}$$

Un cambiamento analogo si può fare per il local best PSO nella (3.3).

Il valore di w influenza fortemente la convergenza dell'algoritmo. Se $w \geq 1$ le velocità aumentano e lo sciame diverge. Se invece $w < 1$, le particelle "decelerano" finchè la loro velocità non diventa nulla. Pertanto valori vicini a uno favoriscono l'esplorazione aumentando la diversità dello sciame; mentre valori più piccoli consento all'algoritmo di concentrarsi nelle zone più promettenti

dello spazio di ricerca. È chiaro che più w è piccolo più la componente sociale e cognitiva, influiscono sull'aggiornamento della posizione. Il valore ottimale di w dipende dal problema e va scelto in funzione dei coefficienti di accelerazione c_1, c_2. Per garantire la convergenza [vdBE06] è necessario che:

$$w > \frac{1}{2}(c_1 + c_2) - 1$$

se questa condizione non è soddisfatta lo sciame potrebbe avere un comportamento divergente o ciclico.

In genere il fattore d'inerzia viene fatto variare dinamicamente man mano che il processo di ottimizzazione procede. Tipicamente si usano due modelli per modificarne il valore:

1. Modifiche casuali. In questo caso in ogni iterazione viene scelto un fattore diverso per esempio campionando da una distribuzione Gaussiana:

   $$w \sim N(0.72, \sigma),$$

   dove σ è piccolo abbastanza da garantire che w non sia troppo più grande di uno.

2. Decremento lineare. In questo caso si parte da un valore prossimo a uno (in genere 0.9) che viene poi decrementato linearmente ad ogni passo temporale t:

   $$w^{(t)} = (w^{(0)} - w^{(t_{max})})\frac{(t_{max} - t)}{t_{max}} + w^{(t_{max})}.$$

   dove t_{max} è il massimo di iterazioni eseguite dall'algoritmo e $w^{(t_{max})}$ è il fattore d'inerzia finale. Chiaramente si ha che $w^{(0)} > w^{(t_{max})}$.

3.2.3 Fattore di costrizione

Un approccio molto simile al precedente è quello in cui le velocità delle particelle sono limitate da una costante χ, chiamata fattore di costrizione. Utilizzando

questo l'equazione di aggiornamento della velocità cambia come segue:

$$v_{i,j}^{(t+1)} = \chi \left[v_{i,j}^{(t)} + c_1 r_{1,j} \left(y_{i,j}^{(t)} - x_{i,j}^{(t)} \right) + c_2 r_{2,j} \left(\hat{y}_j^{(t)} - x_{i,j}^{(t)} \right) \right], \tag{3.5}$$

dove

$$\chi = \frac{2k}{\left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|}, \tag{3.6}$$

e $\phi = c_1 + c_2$.

Se $\phi \geq 4$ e $k \in [0, 1]$ lo sciame converge verso un punto stabile. Il fattore di costrizione, χ, ha un valore compreso nell'intervllo $[0, 1]$, cosicchè la velocità è ridotta ad ogni passo. Il parametro k, tipicamente impostato ad un valore costante, controlla la capacità esplorativa dello sciame. Se $k \approx 1$ l'algoritmo converge lentamente ed ha un altro grado di esplorazione. Viceversa se $k \approx 0$ lo sciame converge rapidamente al costo di una minore esplorazione.

Come accennato in precedenza il modello del fattore di costrizione è molto simile a quello del fattore d'inerzia, a tal punto che per un dato χ, il modello equivalente può essere ottenuto mediante semplici sostituzioni. Per esempio imponendo $c_1 = c_2 = 2.05$, $k = 1$ e sostituendo $\phi = c_1 + c_2 = 4.1$ nella (3.6), si ottiene che $\chi = 0.729$. Se si sostituiscono questi valori nella (3.5) si ha che:

$$v_{i,j}^{(t+1)} = 0.729 \left[v_{i,j}^{(t)} + 2.05 \times r_{1,j} \left(y_{i,j}^{(t)} - x_{i,j}^{(t)} \right) + 2.05 \times r_{2,j} \left(\hat{y}_j^{(t)} - x_{i,j}^{(t)} \right) \right],$$

ora poichè $2.05 \times 0.729 = 1.495$ questo equivale ad utilizzare i valori $c_1 = c_2 = 1.495$ e $w = 0.729$ nella (3.4).

Il vantaggio principale nell'approccio con il fattore di costrizione è che non è necessario il controllo delle velocità.

4

Algoritmo proposto

Nel Paragrafo 2.3 si è visto che il restauro cieco d'immagine può essere riformulato come un problema di minimizzazione di una funzione non globalmente convessa. In particolare il problema consiste nel trovare quell'operatore di sfocatura A che minimizza la funzione energia E_d nella (2.15).

La minimizzazione di funzioni non lineari e/o non convesse, mediante ottimizzazione a sciame di particelle, è un problema ben consolidato in letteratura [SE98, SE99] e in questo capitolo vedremo come un algoritmo di questo tipo è stato implementato per risolvere il problema in questione.

4.1 Scelte implementative

L'algoritmo si basa sulla versione originale del global best PSO, con alcune modifiche e miglioramenti. Le caratteristiche fondamentali sono le seguenti:

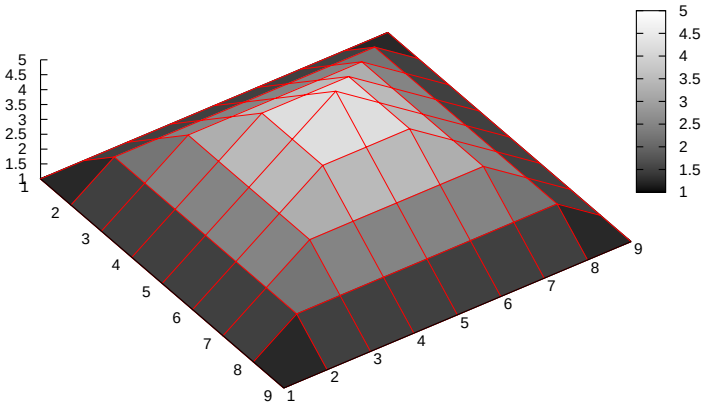
- **Dimensione della popolazione.** Il numero di individui all'interno della popolazione influenza la diversità iniziale dello sciame. Maggiore è il numero di particelle, maggiore è la copertura dello spazio di ricerca ad ogni iterazione. La dimensione della popolazione tuttavia influisce anche sulla complessità dell'algoritmo dal momento che ad ogni passo temporale ogni individuo deve valutare la funzione di fitness. Pertanto lavorare con un numero elevato di particelle è improponibile nei casi in cui, come il nostro, la funzione obiettivo è particolarmente dispendiosa.

Per raggiungere un buon compromesso tra copertura dello spazio e velocità di calcolo si è scelto, come suggerito in [vdBE01], di lavorare con 30 particelle.

- **Codifica ed inizializzazione della posizione.** La posizione di ogni particella specifica un operatore di sfocatura rappresentato dalla rispettiva maschera di blur $M \in \mathbb{R}^{(2h+1)\times(2h+1)}$, $h \in \mathbb{N}^+$, in forma lessicografica. Questa codifica è abbastanza immediata se si considera il fatto che ogni elemento, $m_{i,j}$, della matrice M, fornisce una sorta di informazione spaziale indipendente l'una dall'altra (vedi Figura 4.1).

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 4 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 4 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

(a)



(b)

Figura 4.1: Un esempio di maschera di blur 9×9 (a) e la sua rappresentazione grafica (b).

Si assume che gli elementi della matrice originale M^*, siano tutti compresi nell'intervallo continuo $[0, \mu]$ cosicchè, la posizione iniziale di ogni particella i per ogni dimensione j è data da:

$$x_{i,j} = r_j \mu, \quad \forall j = 1, \ldots, (2h+1)^2, \quad \forall i = 1, \ldots, s \tag{4.1}$$

con $s \in \mathbb{N}^+$ uguale alla dimensione dello sciame e $r_j \sim U(0,1)$.

- **Fitness.** La funzione obiettivo è la seguente:

$$f(A) = E_d(A, \vec{\mathbf{x}}(A)),$$

dove $\vec{\mathbf{x}}(A)$, definito nell'equazione (2.16), corrisponde al risultato del problema inverso associato all'operatore di sfocatura A ed è qui approssimato dall'algoritmo 2.2 con input $(A, \vec{\mathbf{y}})$. Se si considera che la complessità di quest'ultimo è $O(k\lambda N^2)$ (vedi Sottoparagrafo 2.2.2), si capisce come la valutazione della funzione di fitness abbia un costo proibitivo nelle immagini con un numero di pixel (N^2) elevato. Per evitare questo problema si è utilizzata una tecnica, di cui vedremo i dettagli tra breve, che consiste nel valutare la funzione obiettivo in una "sottoimmagine" più piccola, ma comunque rappresentativa dell'originale.

- **Fattore di costrizione.** Per assicurare la convergenza ad un punto stabile si è scelto di controllare le velocità delle particelle utilizzando un fattore di costrizione come in equazione (3.5). Nel nostro caso si è imposto $k = 1$ in modo tale da consentire un alto grado di esplorazione.

- **Coefficienti di accelerazione.** I coefficienti di accelerazione, c_1 e c_2, specificano rispettivamente quanto ogni particella "confida" in se stessa e nel gruppo. Se $c_1 = c_2 = 0$ non viene recepita alcuna informazione nè dall'esperienza personale nè da quella sociale, di conseguenza ogni particella continua a muoversi (ammesso che la velocità iniziale sia diversa da zero) finchè non raggiunge i limiti dello spazio di ricerca. Se $c_1 > 0$ e $c_2 = 0$ ogni particella confida solo in se stessa eseguendo una ricerca

locale. Viceversa se $c_2 > 0$ e $c_1 = 0$ l'intero sciame è attratto verso un singolo punto riducendo quasi a zero la capacità esplorativa. Si è detto (vedi Sottoparagrafo 3.2.3) che quando si usa il fattore di costrizione, è necessario, al fine della convergenza, che la somma della componente sociale e cognitiva sia circa 4. Una scelta tipica è quella di impostare i valori $c_1 = c_2 = 2.05$. Nel nostro caso tuttavia per evitare di far convergere prematuramente lo sciame verso un ottimo locale, si è scelto di imporre $c_1 = 2.8$ e $c_2 = 1.3$.

- **Condizione d'uscita.** L'algoritmo, come vedremo tra poco, estrae in successione una serie di sottoimmagini e lavora su queste. Il restauro coinvolge solo la sottoimmagine interessata e ogni volta che se ne estrae una nuova, la posizione delle particelle viene reinizializzata; mantenendo però la migliore trovata al passo precedente. Il processo in ogni sottoimmagine termina dopo un numero fissato di passi. L'algoritmo invece termina quando la miglior posizione trovata rimane la stessa per un certo numero di estrazioni consecutive o quando il numero massimo di queste è stato raggiunto.

4.2 Confinamento

Il fattore di costrizione non garantisce che la posizione di ogni particella in ogni dimensione j rimanga confinata nell'intervallo $[0, \mu]$. Per fare in modo che questo accada si è implementata una tecnica di confinamento ibrida che modifica le equazioni di aggiornamento della posizione e della velocità. L'idea che sta alla base di questa è che il confinamento può essere fatto in due modi:

1. Limitando la velocità, in modo tale che sia garantito che la nuova posizione cada nell'intervallo prefissato.

2. Vincolando la nuova posizione a stare nei limiti consentiti.

Nel primo caso (vedi Figura 4.2), se si considera una singola particella i, l'equazione di aggiornamento della velocità per ogni dimensione j, può essere riscritta come mostrato nel seguente pseudo-codice:

Algoritmo 4.1 Confinamento iperbolico

for all $j = 1, \ldots, (2h+1)^2$ **do**

$$v_{i,j}^{(t+1)} \leftarrow \chi \left[v_{i,j}^{(t)} + c_1 r_{1,j} \left(y_{i,j}^{(t)} - x_{i,j}^{(t)} \right) + c_2 r_{2,j} \left(\hat{y}_j^{(t)} - x_{i,j}^{(t)} \right) \right]$$

if $v_{i,j}^{(t+1)} > 0$ **then**

$$v_{i,j}^{(t+1)} \leftarrow \frac{v_{i,j}^{(t+1)}}{1 + \left| \dfrac{v_{i,j}^{(t+1)}}{\mu - x_{i,j}^{(t)}} \right|}$$

else

$$v_{i,j}^{(t+1)} \leftarrow \frac{v_{i,j}^{(t+1)}}{1 + \left| \dfrac{v_{i,j}^{(t+1)}}{x_{i,j}^{(t)}} \right|}$$

end if

$$x_{i,j}^{(t+1)} \leftarrow x_{i,j}^{(t)} + v_{i,j}^{(t+1)}$$

end for

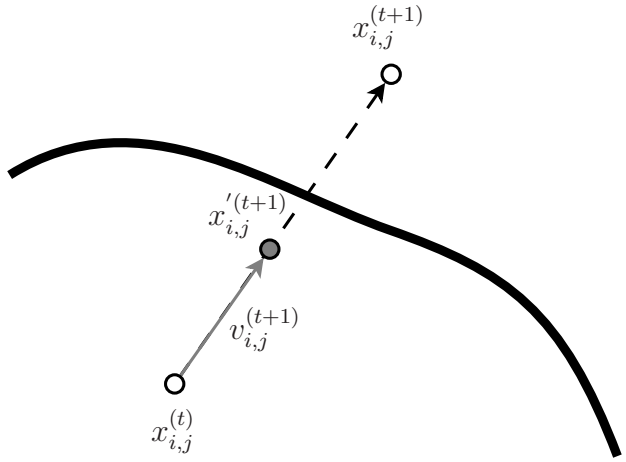


Figura 4.2: Illustrazione del confinamento iperbolico.

Nel secondo invece (vedi Figura 4.3), si agisce direttamente sulla posizione $x_{i,j}^{(t+1)}$ costringendola a rimanere nell'intervallo $[0, \mu]$. Se $x_{i,j}^{(t+1)} > \mu$ si impone che $x_{i,j}^{(t+1)} = \mu$ e si fa in modo che la velocità al passo successivo riporti la particella nello spazio di ricerca, ovvero $v_{i,j}^{(t+1)} = -r_{3,j} v_{i,j}^{(t+1)}$, dove $r_{3,j} \sim U(0,1)$ è un coefficiente casuale diverso per ogni dimensione j. Considerazioni del tutto analoghe valgono nel caso in cui $x_{i,j}^{(t+1)} < 0$.

Lo pseudo-codice 4.2 riassume queste situazioni.

Algoritmo 4.2 Confinamento con ritorno indietro casuale

for all $j = 1, \ldots, (2h+1)^2$ **do**
  $v_{i,j}^{(t+1)} \leftarrow \chi \left[v_{i,j}^{(t)} + c_1 r_{1,j} \left(y_{i,j}^{(t)} - x_{i,j}^{(t)} \right) + c_2 r_{2,j} \left(\hat{y}_j^{(t)} - x_{i,j}^{(t)} \right) \right]$
  $x_{i,j}^{(t+1)} \leftarrow x_{i,j}^{(t)} + v_{i,j}^{(t+1)}$
  if $x_{i,j}^{(t+1)} > \mu$ **then**
    $x_{i,j}^{(t+1)} \leftarrow \mu$
    $v_{i,j}^{(t+1)} \leftarrow -r_{3,j} v_{i,j}^{(t+1)}$
  end if
  if $x_{i,j}^{(t+1)} < 0$ **then**
    $x_{i,j}^{(t+1)} \leftarrow 0$
    $v_{i,j}^{(t+1)} \leftarrow -r_{3,j} v_{i,j}^{(t+1)}$
  end if
end for

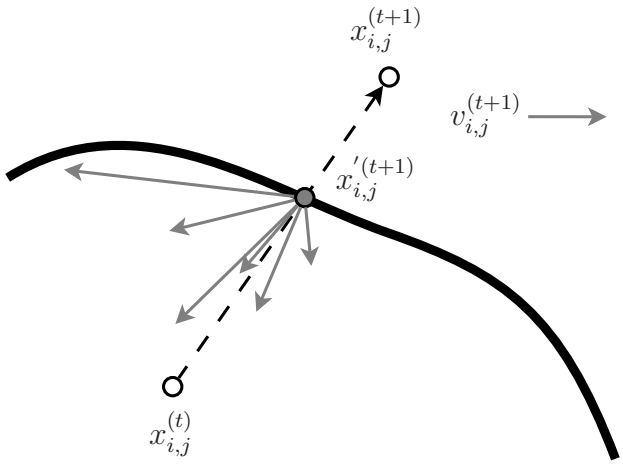


Figura 4.3: Illustrazione del confinamento con ritorno indietro casuale.

L'algoritmo proposto sfrutta ambedue le tecniche: ad ogni passo temporale ogni particella sceglie a caso, con probabilità $\frac{1}{2}$, uno dei due metodi ed aggiorna la propia posizione in funzione di questo. Lo schema che ne deriva è il seguente:

Algoritmo 4.3 Confinamento ibrido

```
for all i ∈ s do
    p ← pseudo-casuale ∈ [0, 1]
    if p ≤ 0.5 then
        aggiorna la velocità e la posizione di i con l'algoritmo 4.1
    else
        aggiorna la velocità e la posizione di i con l'algoritmo 4.2
    end if
end for
```

4.3 Riduzione della complessità della funzione di fitness

L'obiettivo è quello di trovare un metodo che consenta di ridurre il costo computazionale della funzione di fitness mantenendone però l'accuratezza. L'idea che ne sta alla base consiste nell'individuare una stima di $f(A)$ che possa essere calcolata velocemente. La caratteristica fondamentale del metodo utilizzato è che questa stima, non è data da un' approssimazione di $f(A)$ sull'intera immagine, ma dal suo valore calcolato localmente in una sottoarea dell'immagine stessa. La sottoarea individua un'immagine locale più piccola di quella originale in cui tutte le particelle valutano la bontà del propio operatore di sfocatura. Pertanto il primo passo consiste nel poter estrarre dai dati $\vec{\mathbf{y}}$, una sottoimmagine $\hat{\mathbf{y}}$ di dimensione $\hat{N} \times \hat{N}$, $\hat{N} \in \mathbb{N}^+$.

Dal momento che l'algoritmo lavora solo su una sottoimmagine è molto probabile che l'operatore di sfocatura ricavato dal PSO sia buono solo nella zona in considerazione e non sull'immagine intera. Per evitare questo problema la sottoarea viene cambiata periodicamente dopo che le particelle hanno compiuto un numero fissato (ν) di movimenti. Ogni volta che si cambia sottoimmagine lo sciame viene reinizializzato facendo però in modo che una particella abbia

come posizione iniziale la migliore trovata nella zona precedente. In questo modo l'informazione ricavata dall'attività di ottimizzazione passata non va persa e i nuovi individui, di volta in volta, possono disporre dell'esperienza acquisita dai loro predecessori.

Se la sequenza delle zone estratte copre le parti più significative dell'immagine originale, allora il metodo considerato è equivalente a quello che lavora sull'immagine intera. Il vantaggio sta chiaramente nella riduzione della complessità infatti:

1. Non è necessario che la sequenza delle sottoimmagini fornisca una copertura completa.

2. Dal momento che la complessità della funzione di fitness è in gran parte determinata dal termine $\vec{\mathbf{x}}(A)$ che si stima avere un costo computazionale dell'ordine di $O(k\lambda N^2)$, lavorare con una dimensione $\hat{N}$ più piccola significa guadagnare una riduzione quadratica nel tempo di calcolo.

È chiaro che non tutte le sottoimmagini hanno la stessa importanza, infatti una zona totalmente uniforme fornirà al processo di ottimizzazione una informazione completamente differente rispetto a quella data da una caratterizzata da molti bordi. Per fare in modo che tutte quelle estratte siano ugualmente "significative" si utilizza un metodo che ne misura la regolarità (vedi Figura 5.3, 5.5 e 5.7). In particolare una sottoimmagine $\hat{\mathbf{y}}$ di dimensione $\hat{N} \times \hat{N}$ è ricavata a caso dai dati $\vec{\mathbf{y}}$ in modo tale che la sua regolarità $S(\hat{\mathbf{y}})$ cada in un intervallo prefissato $[l, u]$ dove

$$S(\hat{\mathbf{y}}) = \sum_{c \in C_k} g_{\alpha,\lambda}(D_c^k \hat{\mathbf{y}}), \tag{4.2}$$

e $g_{\alpha,\lambda}(D_c^k \hat{\mathbf{y}})$ definita come nell'equazione (2.8). Se i parametri l e u sono scelti in maniera corretta, ogni immagine ricavata da questa procedura sarà ugualmente significativa.

In conclusione l'algoritmo proposto può essere schematizzato come mostrato nello pseudo-codice 4.4. In questo bisogna notare una leggera differenza di notazione. In particolare la miglior posizione associata a ciascuna particella è indicata dal simbolo $\vec{\mathbf{z}}_i$, mentre $\hat{\mathbf{z}}$ identifica la migliore trovata dallo sciame.

Algoritmo 4.4 Image restoration PSO

Require: $\vec{\mathbf{y}}$ l'immagine osservata
Return: $\hat{\mathbf{z}}$ una stima dell'operatore di sfocatura A

repeat
  estrai una sottoimmagine $\hat{\mathbf{y}}$ dai dati $\vec{\mathbf{y}}$
  for all $i \in s$ **do** {inizializzazione}
    if $\hat{\mathbf{y}}$ non è la prima estratta **and** $i = 1$ **then**
      $\vec{\mathbf{x}}_i \leftarrow \hat{\mathbf{z}}$
    else
      inizializza $\vec{\mathbf{x}}_i$ con la (4.1)
    end if
    $\vec{\mathbf{v}}_i \leftarrow 0$
    $\vec{\mathbf{z}}_i \leftarrow \vec{\mathbf{x}}_i$
  end for
  for $n = 1, \ldots, \nu$ **do**
    for all $i \in s$ **do**
      if $f(\vec{\mathbf{x}}_i) < f(\vec{\mathbf{z}}_i)$ **then**
        $\vec{\mathbf{z}}_i \leftarrow \vec{\mathbf{x}}_i$
      end if
      if $f(\vec{\mathbf{z}}_i) < f(\hat{\mathbf{z}})$ **then**
        $\hat{\mathbf{z}} \leftarrow \vec{\mathbf{z}}_i$
      end if
    end for
    aggiorna la velocità e la posizione di ogni i con l'algoritmo 4.3
  end for
until $\hat{\mathbf{z}}$ non è la stessa per t estrazioni consecutive **or** il massimo numero di queste è stato raggiunto

5

Risultati sperimentali

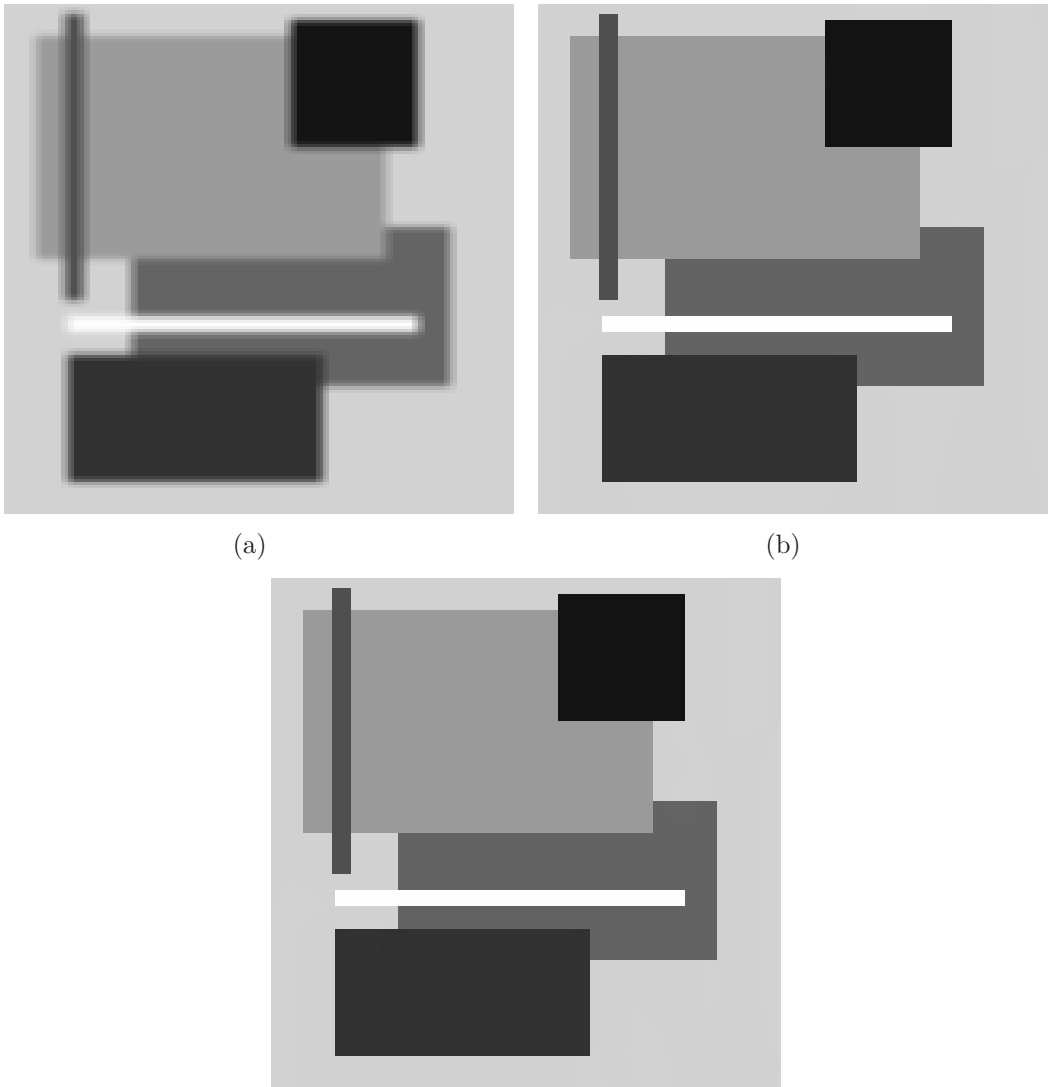
In questo capitolo mostriamo alcuni risultati ottenuti dall'algoritmo appena descritto considerando sia immagini reali che stilizzate.

I parametri α e λ che insieme definiscono la soglia entro cui ricercare le discontinuità, sono stati scelti in maniera empirica basandosi sulla qualità del risultato ottenuto quando si conosce il processo di degradazione (restauro non cieco). Gli stessi valori sono stati poi utilizzati nella funzione di fitness nel caso del restauro cieco. In letteratura esistono comunque diversi metodi per determinarne un stima [BFT91, GMS06, Han92, Reg96]. In ogni test il massimo elemento della matrice di sfocatura è stato fissato a $\mu = 15$ cosicchè la posizione di ogni particella in ogni direzione rimane confinata nell'intervallo $[0, 15]$. La dimensione della sottoimmagine invece varia da caso a caso.

Il primo test è stato fatto su un'immagine 160×160 (visibile in Figura 5.1(a)) caratterizzata da ampie zone regolari, sfocata con la maschera

$$M_1 = \begin{pmatrix} 2 & 1 & 3 & 1 & 2 \\ 4 & 2 & 6 & 2 & 4 \\ 8 & 4 & 12 & 4 & 8 \\ 4 & 2 & 6 & 2 & 4 \\ 2 & 1 & 3 & 1 & 2 \end{pmatrix}.$$

La Figura 5.1(b) mostra il risultato del restauro non cieco effettuato con clique del primo ordine e i parametri $\alpha = 400$ e $\lambda = 2$. Nella 5.1(c) invece è illustrato il risultato di quello cieco, ricavato, date le dimensioni ridotte e la particolare struttura geometrica, lavorando direttamente sull'intera immagine.



(a) (b)

(c)

Figura 5.1: Test 1: (a) immagine osservata, (b) restauro non cieco, (c) restauro cieco.

Nel secondo test si è utilizzata una foto ricca di contenuti nota in letteratura col nomignolo "Lena" dal nome dell'ex modella svedese, Lena Sjööblom, in questa rappresentata. L'immagine sfocata con la maschera

$$M_2 = \begin{pmatrix} 1 & 1 & 2 & 3 & 2 & 1 & 1 \\ 2 & 2 & 4 & 6 & 4 & 2 & 2 \\ 3 & 3 & 6 & 9 & 6 & 3 & 3 \\ 4 & 4 & 8 & 12 & 8 & 4 & 4 \\ 3 & 3 & 6 & 9 & 6 & 3 & 3 \\ 2 & 2 & 4 & 6 & 4 & 2 & 2 \\ 1 & 1 & 2 & 3 & 2 & 1 & 1 \end{pmatrix},$$

è visibile in Figura 5.2(a). Il risultato del restauro non cieco ottenuto utilizzando clique del secondo ordine e i parametri $\alpha = 4$ e $\lambda = 2$ è mostrato nella 5.2(b). Per quanto riguarda il restauro cieco invece si è fissata la dimensione della sottoimmagine a $\hat{N} = 100$ e i limiti per l'estrazione di quest'ultima sono stati scelti in modo tale da escludere regioni troppo e troppo poco regolari. In particolare nelle zone caratterizzate da una forte uniformità, la differenza del valore di fitness associato a diverse maschere di sfocatura non è apprezzabile, cosicchè il processo di ottimizzazione può essere compromesso. In quelle troppo poco regolari invece è possibile che la ricchezza di bordi venga confusa con il rumore. Dopo aver studiato la composizione delle regioni coperte dalla procedura di estrazione della sottoimmagine (vedi Figura 5.3), si è imposto come limite inferiore e superiore rispettivamente $l = 51000$ e $u = 52000$. Il risultato dato dall'algoritmo proposto è quello mostrato in Figura 5.2(c).

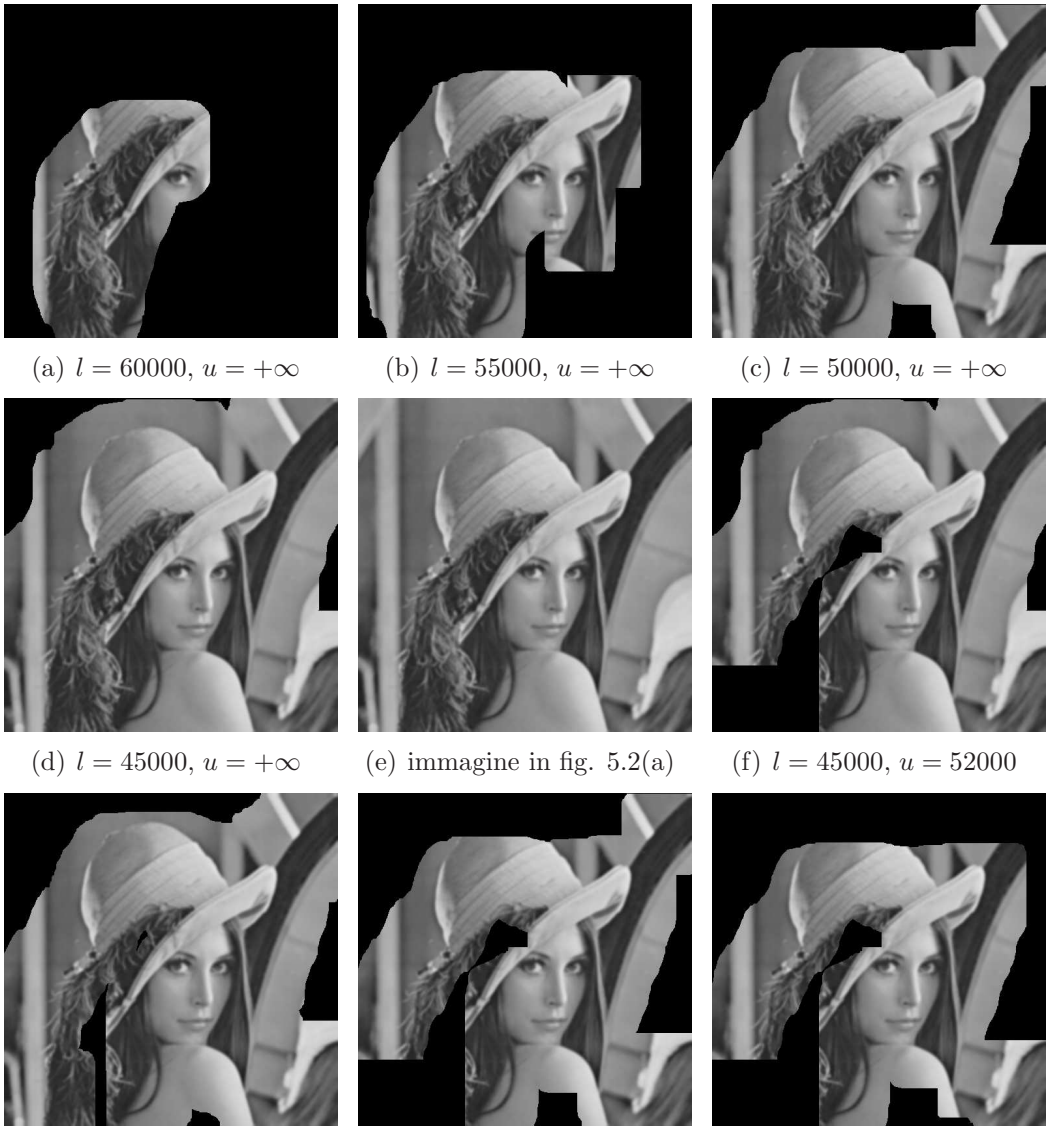
Dal momento che la struttura della "Lena" è particolarmente complicata, si è cercato di migliorare la qualità del risultato del restauro cieco, facendo delle ipotesi sulla geometria della maschera di sfocatura con l'obiettivo di ridurre lo spazio di ricerca. Si può vedere che M_2 è simmetrica e separabile, nel senso che può essere scomposta nel prodotto di due vettori: uno colonna $\vec{\mathbf{c}} = (1, 2, 3, 4, 3, 2, 1)$ e uno riga $\vec{\mathbf{d}} = (1, 1, 2, 3, 2, 1, 1)$. Questi sono a loro a volta simmetrici, pertanto se la dimensione della maschera è $(2h + 1) \times (2h + 1)$, l'informazione significativa è contenuta solo nella prime (o equivalentemente

nelle ultime) $h + 1$ componenti. Si è quindi provato a codificare la posizione di ogni particella utilizzando due vettori di questo tipo inizializzati con valori a caso nell'intervallo $[0, \mu]$, ricavando poi da questi la corrispondente matrice necessaria per valutare la funzione di fitness. Il risultato che sfrutta gli stessi parametri del caso precedente è mostrato in Figura 5.2(d).

(a) (b)

(c) (d)

Figura 5.2: Test 2: (a) immagine osservata, (b) restauro non cieco, (c) restauro cieco, (d) restauro cieco con una diversa codifica della posizione delle particelle.



(a) $l = 60000$, $u = +\infty$ (b) $l = 55000$, $u = +\infty$ (c) $l = 50000$, $u = +\infty$

(d) $l = 45000$, $u = +\infty$ (e) immagine in fig. 5.2(a) (f) $l = 45000$, $u = 52000$

(g) $l = 48000$, $u = 54000$ (h) $l = 50000$, $u = 52000$ (i) $l = 51000$, $u = 52000$

Figura 5.3: Parti della foto 5.2(a) coperte dalla procedura di estrazione della sottoimmagine con diversi valori di l e u.

Il terzo esperimento è stato fatto con la foto mostrata in Figura 5.4(a) sfocata con la maschera

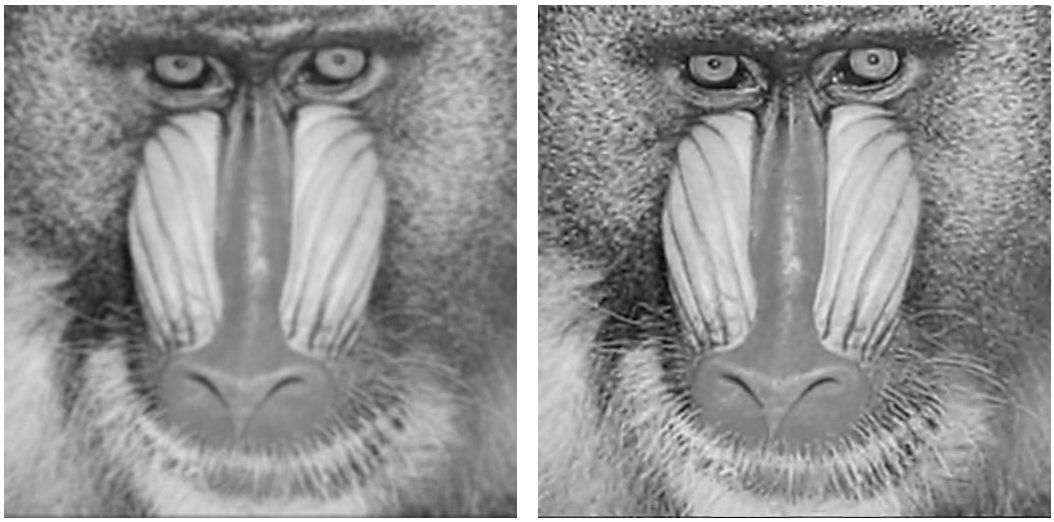
$$M_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 3 & 4 & 5 & 4 & 3 & 2 \\ 3 & 4 & 5 & 6 & 5 & 4 & 3 \\ 4 & 5 & 6 & 7 & 6 & 5 & 4 \\ 3 & 4 & 5 & 6 & 5 & 4 & 3 \\ 2 & 3 & 4 & 5 & 4 & 3 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

L'immagine 5.4(b) riporta il risultato dell'algoritmo 2.2 avviato con l'operatore M_3, $\alpha = 4$, $\lambda = 0.5$ e clique del secondo ordine. La scelta dei limiti $l = 24000$ e $u = 27000$ (vedi Figura 5.5), è stata dettata da considerazioni del tutto analoghe a quelle fatte in precedenza e la dimensione della sottoimmagine è stata fissata a $\hat{N} = 100$. Il risultato del restauro cieco ottenuto con questa configurazione è visibile in Figura 5.4(c).

L'ultima immagine presentata è quella in Figura 5.6(a) corrotta da

$$M_4 = \begin{pmatrix} 2 & 3 & 4 & 3 & 2 \\ 4 & 6 & 8 & 6 & 4 \\ 6 & 9 & 12 & 9 & 6 \\ 4 & 6 & 8 & 6 & 4 \\ 2 & 3 & 4 & 3 & 2 \end{pmatrix}$$

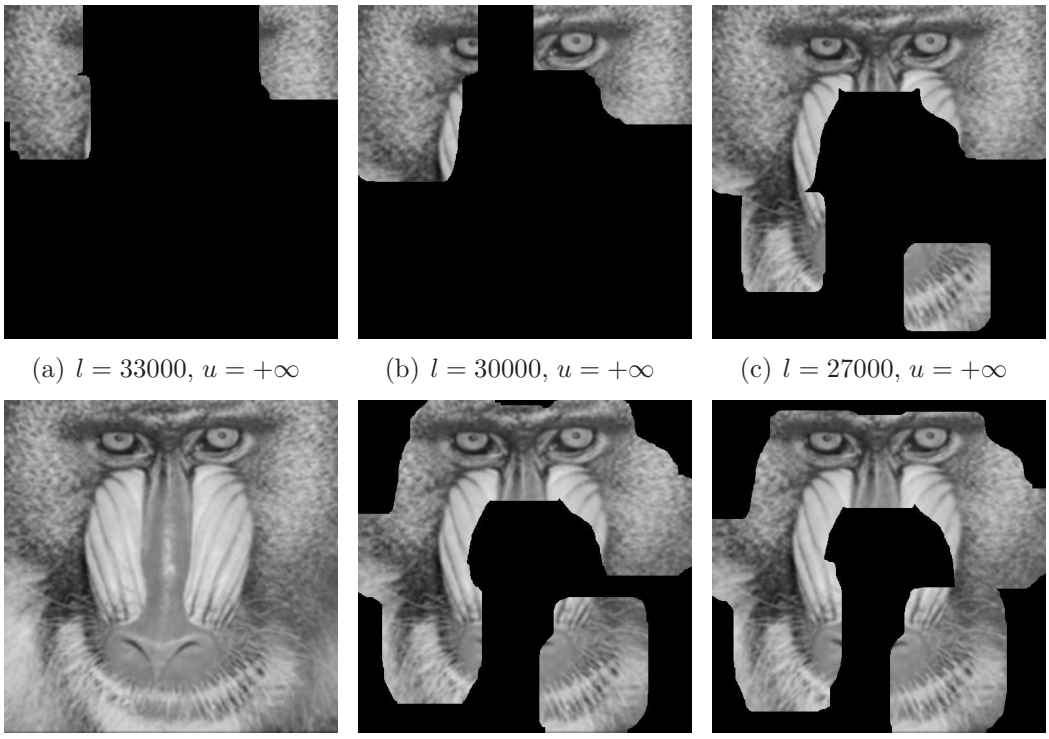
e da un rumore aggiuntivo con varianza $\sigma^2 = 100$. Il restauro non cieco, mostrato in Figura 5.6(b), è stato ottenuto utilizzando clique del secondo ordine e i parametri $\alpha = 625$ e $\lambda = 8$. Per quanto riguarda quello cieco invece, si è fissata la dimensione della sottoimmagine a $\hat{N} = 50$ e si sono scelti i limiti $l = 2760000$ e $u = 2783000$ cercando ancora una volta un compromesso tra zone particolareggiate e uniformi. In questo caso (vedi Figura 5.7) bisogna notare una frammentazione, dovuta al rumore, nelle aree coperte dalla procedura di estrazione della sottoimmagine. Il risultato ottenuto con l'algoritmo proposto è mostrato in Figura 5.6(c).



(a) (b)

(c)

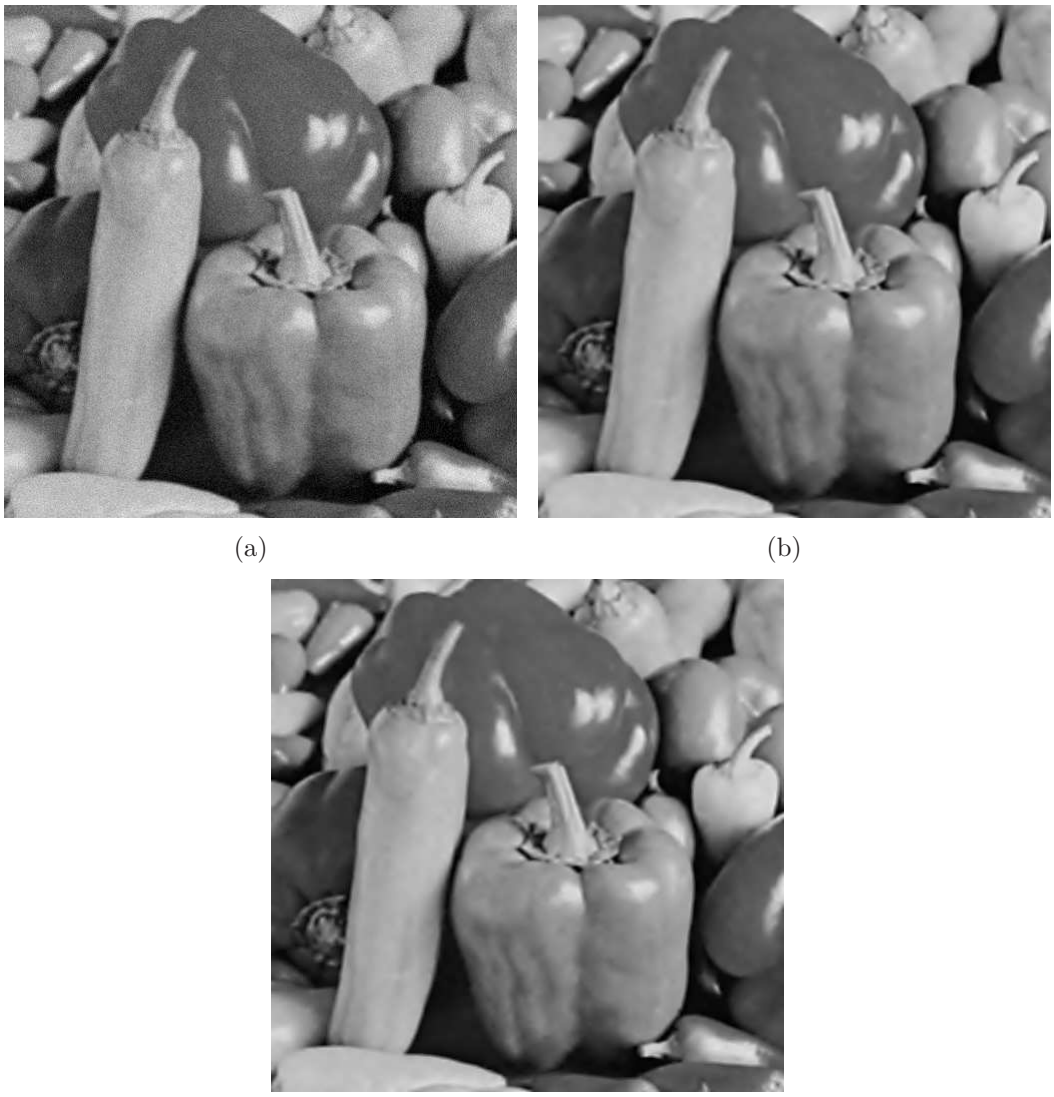
Figura 5.4: Test 3: (a) immagine osservata, (b) restauro non cieco, (c) restauro cieco.



(a) $l = 33000$, $u = +\infty$ (b) $l = 30000$, $u = +\infty$ (c) $l = 27000$, $u = +\infty$

(d) immagine in fig. 5.4(a) (e) $l = 25000$, $u = 28000$ (f) $l = 24000$, $u = 27000$

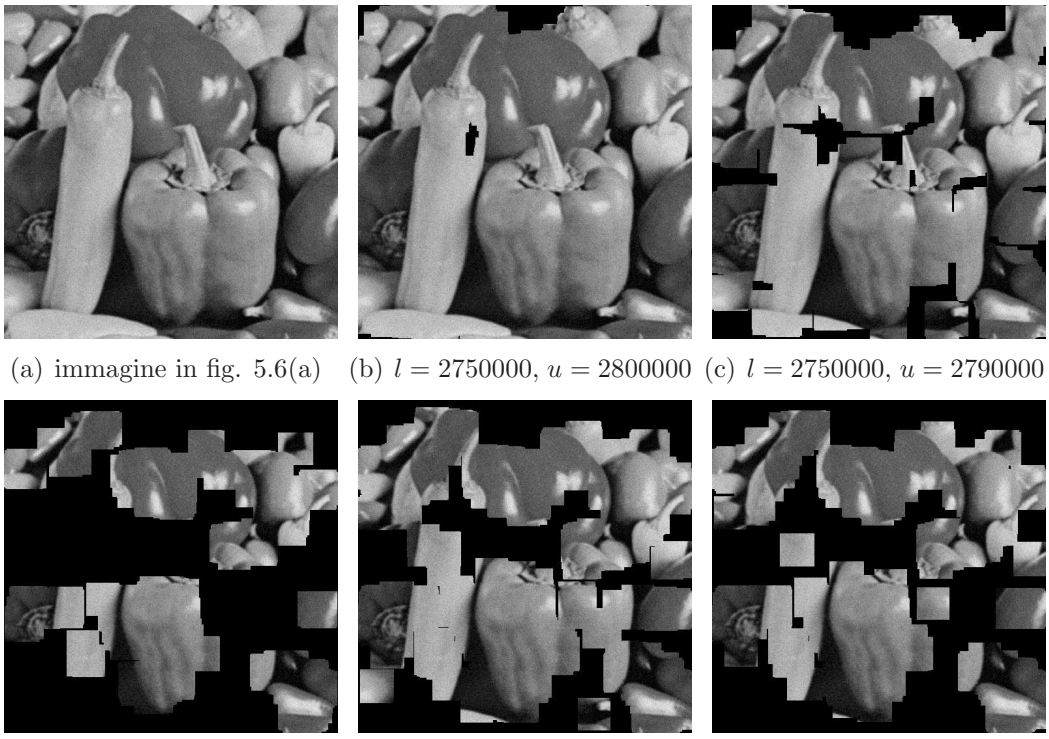
Figura 5.5: Parti della foto 5.4(a) coperte dalla procedura di estrazione della sottoimmagine con diversi valori di l e u.



(a) (b)

(c)

Figura 5.6: Test 4: (a) immagine osservata, (b) restauro non cieco, (c) restauro cieco.



(a) immagine in fig. 5.6(a) (b) $l = 2750000$, $u = 2800000$ (c) $l = 2750000$, $u = 2790000$

(d) $l = 2760000$, $u = 2780000$ (e) $l = 2760000$, $u = 2785000$ (f) $l = 2760000$, $u = 2783000$

Figura 5.7: Parti della foto 5.6(a) coperte dalla procedura di estrazione della sottoimmagine con diversi valori di l e u.

Conclusioni

In questa tesi si è implementato un algoritmo evolutivo per risolvere il problema del restauro cieco d'immagine. La tecnica proposta coincide con una versione ibrida dell'ottimizzazione a sciame di particelle e si basa su un concetto di selezione tipico degli algoritmi genetici. Questo consente al PSO di lavorare solo su una sottoimmagine che periodicamente viene cambiata. Lo sciame, di volta in volta reinizializzato, mantiene ad ogni generazione (secondo la regola della selezione "elitaria") il miglior individuo della popolazione precedente. In questo modo si crea una forma di apprendimento automatico in cui le informazioni rilevanti trovate in una sottoimmagine sono trasmesse di generazione in generazione.

Il vantaggio principale di questa tecnica è che da un lato permette di ridurre la complessità computazionale, dall'altro di ottenere una maggiore diversità dello sciame che si traduce in una migliore copertura dello spazio di ricerca. Risultati soddisfacenti sono stati ottenuti in termini di qualità delle immagini restaurate come mostrato nel Capitolo 5.

A

Derivate parziali delle funzioni approssimanti

Si è detto nel Sottoparagrafo 2.2.2 che le varie funzioni approssimanti della (2.7), al variare del parametro $p \in \mathbb{R}, p \in (0, 2]$, sono costruite modificando solamente la parte relativa alla funzione di interazione g. Ovvero

$$E_d^{(p)}(A, \vec{\mathbf{x}}) = ||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 + \sum_{c \in C_k} g^{(p)}(D_c^k \vec{\mathbf{x}}). \tag{A.1}$$

Calcoliamo qui le derivate parziali prime e seconde di queste.

A.1 Derivate parziali prime

Consideriamo prima solo la parte relativa al termine $||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2$ indipendente dal parametro p. Chiamiamo questa $\hat{E}_d(A, \vec{\mathbf{x}})$.

La norma euclidea di un vettore $\vec{\mathbf{x}}$ è data da $||\vec{\mathbf{x}}|| = \sqrt{\sum_{i=1}^{n} x_i^2}$. Quindi

$$\begin{aligned}
\vec{\mathbf{y}} - A\vec{\mathbf{x}} &= \begin{pmatrix} y_1 - \sum_{j=1}^{n} a_{1,j}x_j \\ y_2 - \sum_{j=1}^{n} a_{2,j}x_j \\ \vdots \\ y_n - \sum_{j=1}^{n} a_{n,j}x_j \end{pmatrix} \\
&\Downarrow \\
||\vec{\mathbf{y}} - A\vec{\mathbf{x}}||^2 &= \sum_{k=1}^{n} \left(y_k - \sum_{j=1}^{n} a_{k,j}x_j \right)^2 .
\end{aligned}$$

Se si deriva rispetto un x_i si ha:

$$\begin{aligned}
\frac{\partial \hat{E}_d(A, \vec{\mathbf{x}})}{\partial x_i} &= \sum_{k=1}^{n} -2a_{k,i} \left(y_k - \sum_{j=1}^{n} a_{k,j}x_j \right) = \\
&= -2\sum_{k=1}^{n} a_{k,i}y_k + 2\sum_{k=1}^{n} a_{k,i} \sum_{j=1}^{n} a_{k,j}x_j .
\end{aligned} \tag{A.2}$$

Il gradiente di $\hat{E}_d(A, \vec{\mathbf{x}})$ è quindi definito come:

$$\begin{aligned}
\nabla \hat{E}_d &= -2 \begin{pmatrix} \sum_{k=1}^{n} a_{k,1}y_k \\ \sum_{k=1}^{n} a_{k,2}y_k \\ \vdots \\ \sum_{k=1}^{n} a_{k,n}y_k \end{pmatrix} + 2 \begin{pmatrix} \sum_{k=1}^{n} a_{k,1} \sum_{j=1}^{n} a_{k,j}x_j \\ \sum_{k=1}^{n} a_{k,2} \sum_{j=1}^{n} a_{k,j}x_j \\ \vdots \\ \sum_{k=1}^{n} a_{k,n} \sum_{j=1}^{n} a_{k,j}x_j \end{pmatrix} = \\
&= -2A^T\vec{\mathbf{y}} + 2A^T A\vec{\mathbf{x}}.
\end{aligned}$$

Prendiamo ora invece, la parte relativa al termine $\sum_{c \in C_k} g^{(p)}(D_c^k \vec{\mathbf{x}})$ e chiamia-

mola $\bar{E}_d^{(p)}(A, \vec{\mathbf{x}})$. Derivando rispetto un x_i si ha che:

$$\frac{\partial \bar{E}_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i} = \sum_{c \in C_k} g^{'(p)}(D_c^k \vec{\mathbf{x}}) \frac{\partial D_c^k \vec{\mathbf{x}}}{\partial x_i}, \tag{A.3}$$

dove $g^{'(p)}(t)$ indica la derivata prima dell'approssimante $g^{(p)}$ di g e $D_c^k \vec{\mathbf{x}}$ l'operatore differenza finito del vettore $\vec{\mathbf{x}}$ sulla clique c di ordine k (vedi paragrafo 2.1).

Riprendendo la (2.10) per $p = 2$

$$g^{'*}(t) = g^{'(2)}(t) = \begin{cases} 2\lambda^2 t & \text{se } |t| < q(1), \\ 2\lambda^2 q(1) & \text{se } |t| \geq q(1) \text{ e } t \geq 0, \\ -2\lambda^2 q(1) & \text{se } |t| \geq q(1) \text{ e } t < 0. \end{cases}$$

Per $p \in (0, 1]$ dalla (2.9)

$$g^{'(p)}(t) = \begin{cases} 2\lambda^2 t & \text{se } |t| < q(p), \\ -\tau(p)\,(t - r(p)) & \text{se } q(p) \leq |t| \leq r(p) \text{ e } t \geq 0, \\ \tau(p)\,(-t - r(p)) & \text{se } q(p) \leq |t| \leq r(p) \text{ e } t < 0, \\ 0 & \text{se } |t| > r(p). \end{cases}$$

Infine per $p \in (1, 2]$ dalla (2.11)

$$g^{'(p)}(t) = (p-1) g^{'*}(t) + (2-p) g^{'(1)}(t).$$

Pertanto fissato un x_i

$$\frac{\partial E_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i} = \frac{\partial \hat{E}_d(A, \vec{\mathbf{x}})}{\partial x_i} + \frac{\partial \bar{E}_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i}. \tag{A.4}$$

A.2 Derivate parziali seconde

Derivando la (A.4) rispetto x_j si ha che:

$$\frac{\partial^2 E_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i \partial x_j} = \frac{\partial^2 \hat{E}_d(A, \vec{\mathbf{x}})}{\partial x_i \partial x_j} + \frac{\partial^2 \bar{E}_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i \partial x_j}$$

con

$$\frac{\partial^2 \hat{E}_d(A, \vec{\mathbf{x}})}{\partial x_i \partial x_j} = \sum_{k=1}^{n} -2a_{k,i}\,(-a_{k,j}) = 2\sum_{k=1}^{n} a_{k,i}a_{k,j},$$

(chiaramente quando $j = i$)

$$\frac{\partial^2 \hat{E}_d(A, \vec{\mathbf{x}})}{\partial x_i^2} = \sum_{k=1}^{n} -2a_{k,i}\,(-a_{k,j}) = 2\sum_{k=1}^{n} a_{k,i}^2,$$

e

$$\frac{\partial^2 \bar{E}_d^{(p)}(A, \vec{\mathbf{x}})}{\partial x_i \partial x_j} = \sum_{c \in C_k} g^{''(p)}(D_c^k \vec{\mathbf{x}}) \frac{\partial D_c^k \vec{\mathbf{x}}}{\partial x_j} \frac{\partial D_c^k \vec{\mathbf{x}}}{\partial x_i} + g^{'(p)}(D_c^k \vec{\mathbf{x}}) \frac{\partial^2 D_c^k \vec{\mathbf{x}}}{\partial x_i \partial x_j},$$

dove per $p = 2$

$$g^{''*}(t) = g^{''(2)}(t) = \begin{cases} 2\lambda^2 & \text{se } |t| < q(1), \\ 0 & \text{altrimenti,} \end{cases}$$

per $p \in (0, 1]$

$$g^{''(p)}(t) = \begin{cases} 2\lambda^2 & \text{se } |t| < q(p), \\ -\tau(p) & \text{se } q(p) \leq |t| \leq r(p), \\ 0 & \text{se } |t| > r(p), \end{cases}$$

per $p \in (1, 2]$

$$g^{''(p)}(t) = (p-1)g^{''*}(t) + (2-p)g^{''(1)}(t).$$

Bibliografia

[BFT91] L. Bedini, E. Fantini, and A. Tonazzini. A dual approach to regularization in image restoration. *Pattern Recognition Letters*, 1991.

[BGT94] L. Bedini, I. Gerace, and A. Tonazzini. A deterministic algorithm for reconstructing images with interacting discontinuities. *CVGIP. Graphical models and image processing*, 1994.

[Bla89] A. Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1989.

[BPT88] M. Bertero, T. Poggio, and V. Torre. Ill-posed roblems in early vision. *Proceedings of the IEEE*, 1988.

[bz87] *Visual reconstruction*. MIT Press, 1987.

[Dem89] G. Demoment. Image reconstruction and restoration: overview of common estimationstructures and problems. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1989.

[eds96] *Computational intelligence PC tools*. Academic Press Professional, Inc., 1996.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984.

[GMMM08] I. Gerace, M. Mastroleo, A. Milani, and S. Moraglia. Genetic blind image restoration with dynamical local evaluation. *ICCSA '08: Proceedings of the 2008 International Conference on Computational Sciences and Its Applications*, 2008.

[GMS06] I. Gerace, F. Martinelli, and G. Sanchini. Estimation of the free parameters in the problem of edge-preserving image reconstruction by a shooting method. *Proceedings of The 2006 International TICSP Workshop on Spectral Methods and Multirate Signal Processing, SMMSP2006*, 2006.

[GPPS08] I. Gerace, L. Pinca, P. Pucci, and G. Sanchini. Surface image reconstruction for the comet assay technique. *International Journal of Signal and Imaging Systems Engineering*, 2008.

[GR92] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[Han92] Per C. Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review*, 1992.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 1995.

[KGV83] S. Kirkpatrick, C. Gellatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 1983.

[KVR02] T. Krink, J. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. *Proceedings of the Fourth Congress on Evolutionary Computation*, 2002.

[MRR+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 1953.

[Nik99] M. Nikolova. Markovian reconstruction using a gnc approach. *IEEE Transactions on Image Processing*, 1999.

[Reg96] T. Reginska. A regularization parameter in discrete ill-posed problems. *SIAM Journal on Scientific Computing*, 1996.

[SBP83] W. Smith, H. Barrett, and R. Paxman. Reconstruction of objects from coded images by simulated annealing. *Optics Letters*, 1983.

[SE98] Y. Shi and R. Eberhart. A modified particle swarm optimizer. *Proceedings of the 1998 IEEE International Conference of Evolutionary Computation*, 1998.

[SE99] Y. Shi and R. Eberhart. Empirical study of particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999.

[Ton01] A. Tonazzini. Blur identification analysis in blind image deconvolution using markov random fields. *Pattern Recognition and Image Analysis*, 2001.

[vdBE01] F. van den Bergh and A. Engelbrecht. Effects of swarm size on cooperative particle swarm optimisers. *Proceedings of the Genetic and Evoluationary Computation Conference*, 2001.

[vdBE06] F. van den Bergh and A. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 2006.