

UNIVERSITÀ DEGLI STUDI DI CATANIA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

CORSO DI LAUREA IN INFORMATICA

ALESSANDRA LOMBARDO

DATA DRIVEN ZOOMING

UNA TECNICA DI ZOOMING BASATA SU DATA DEPENDENT TRIANGULATION

—————
TESI DI LAUREA
—————

Relatore:

Chiar.mo Prof. GIOVANNI GALLO

Correlatore:

Dott. GIUSEPPE MESSINA

ANNO ACCADEMICO 2002 - 2003

Indice

Introduzione..... pag. 1

Capitolo 1 : Zooming

- 1.1 Risoluzione..... pag. 3
- 1.2 Interpolazione..... pag. 4
- 1.3 Espansione..... pag. 6
- 1.4 Lo stato dell'arte..... pag. 8

Capitolo 2 : Triangolazione Dipendente dai dati (DDT)

- 2.1 La Triangolazione – Definizione del problema. pag. 14
- 2.2 La Triangolazione di Delaunay. pag. 17
- 2.3 Data Dependent Triangulation. pag. 21
- 2.4 Criteri di Ottimalità. pag. 23
- 2.5 Funzioni-costo. pag. 25
 - 2.5.1.Funzioni NC1. pag. 26

Capitolo 3 : Data Driven Zooming (DDZ)

- 3.1 Introduzione..... pag. 28
- 3.2 Strutture Dati..... pag. 31
- 3.3 Funzione costo..... pag. 33
- 3.4 Il corpo dell'algoritmo..... pag. 34
 - 3.4.1 Preprocessing..... pag. 34
 - 3.4.2 Fase Iterativa..... pag. 37
 - 3.4.3 Ottimizzazione..... pag. 39
- 3.5 PSNR..... pag. 41
- 3.6 Immagini a colori..... pag. 44

Capitolo 4 : Sperimentazione

- 4.1 Categorie e dimensione del DataBase..... pag. 45
- 4.2 Analisi dei risultati del testing..... pag. 46
 - 4.2.1 DDZ – andamento PSNR durante le iterazioni.... pag. 47
 - 4.2.2 Statistiche PSNR di vari algoritmi di Zooming... pag. 49

Bibliografia..... pag. 55

Introduzione

Lo Zooming è il processo di ingrandimento delle immagini digitali. A differenza dello zoom dei dispositivi analogici, si tratta di un procedimento software, che aumenta artificialmente la risoluzione spaziale di un file grafico, limitata dalle caratteristiche tecniche del dispositivo di acquisizione digitale, in particolare, dal numero dei suoi fotorivelatori.

Negli ultimi decenni, il problema dello Zooming digitale è stato ampiamente indagato, il modo tipico di procedere consiste nell'espandere la dimensione dell'immagine (cfr. paragrafo 1.1), e poi approssimare i pixel mancanti, per interpolazione.

La tesi qui presentata adopera un tipo particolare di interpolazione, detta *piecewise* (a pezzi), cioè ottenuta suddividendo il piano dell'immagine in sottoaree, tra loro disgiunte a due a due, e ricostruendole separatamente.

Una peculiarità della tecnica usata, è il vincolo di forma triangolare delle sottoaree, (si dice che il piano viene "triangolato").

Altra importante caratteristica è che il criterio di scelta della triangolazione è legato ai valori di colore del particolare input, questo è il principio che sta alla base della *Data Dependent Triangulation* (DDT).

Per questa tesi è stata realizzata un'implementazione software della tecnica di Zooming mediante *Data Dependent Triangulation* presentata, nonché una serie di altre funzionalità ad essa correlate. L'applicazione si chiama DDZ, il file eseguibile è allegato alla tesi in formato compresso, "ddz.zip".

Si è dato un contributo originale al preprocessing e all'ottimizzazione

dell'algoritmo, lo si è testato su un opportuno database di immagini di varia tipologia, e lo si è confrontato con i metodi standard di interpolazione: Replica, Interpolazione Bilineare, Interpolazione Bicubica; e con alcuni tra i metodi noti più innovativi: L.A.Z.A. (BGS), Zooming di ReST, Adaptive Zooming.

Si è implementato e testato, inoltre, l'algoritmo Pixel Level DDT di Su e Willis [5], e se ne è voluta sperimentare una possibile integrazione con l'algoritmo DDZ; ma questi ultimi test non hanno rilevato miglioramenti.

Nel primo capitolo, si dà una panoramica del problema dello Zooming, e dei modi di affrontarlo, il paragrafo 1.3 spiega la procedura di espansione, comune a tutti gli approcci al problema, i più accreditati dei quali, sono presentati nel paragrafo 1.4.

Il secondo capitolo tratta il problema di Geometria Computazionale della Triangolazione; il paragrafo 2.1 lo definisce formalmente, il paragrafo 2.2 ne descrive la soluzione standard: la triangolazione di Delaunay, mentre, nel resto del capitolo, si presenta la triangolazione dipendente dai dati (DDT), sulla quale è basato il metodo Data Driven Zooming (DDZ), oggetto della tesi. Il terzo capitolo è una descrizione dettagliata dell'applicazione DDZ, dalle strutture dati (3.2), alle funzioni costo usate per la scelta della triangolazione ottimale(3.3), all'algoritmo DDZ e la sua ottimizzazione(3.4). Il paragrafo 3.5 introduce la misura di qualità che è stata utilizzata nel testing, e fornita nel programma. Il paragrafo 3.6 parla di come si è affrontato il problema per le immagini a colori e per quelle a toni di grigio.

Al capitolo 4, sono riportati i risultati statistici della sperimentazione svolta: Lo studio riguarda sia il comportamento delle singole fasi iterative dell'algoritmo, sia il confronto con gli altri metodi di Zooming.

Capitolo 1 : Zooming

1.1 Risoluzione

Le informazioni visive che l'occhio umano riceve, sottoforma di energia luminosa, vengono convertite, al suo interno, in informazioni elettriche e chimiche. Sebbene la nostra retina sia dotata di un numero finito di fotorecettori, la visione che abbiamo di ciò che guardiamo, appare omogenea e continua. Grazie a ciò, il segnale fornito dalla nostra vista può essere considerato analogico.

Anche l'acquisizione di un'immagine su computer comporta, proprio per la sua natura digitale, un campionamento del segnale continuo reale in unità atomiche discrete di colore (i pixel). Possiamo, dunque, pensare all'immagine sintetica come ad una approssimazione discreta di una qualche funzione-colore continua.

Maggiore è il numero di campioni in cui si è in grado di suddividere il segnale, e più dettagliata ne risulterà la rappresentazione; all'aumentare del dettaglio, anche l'aspetto sarà migliore.

Il parametro di qualità che indica il massimo livello di dettaglio distinguibile in un segnale, è la sua Risoluzione Spaziale, che nel seguito della tesi sarà chiamata semplicemente "risoluzione".

La risoluzione rappresenta la densità dei pixel in un'immagine, essa misura, cioè, il numero di pixel in cui è campionata l'unità di superficie dell'immagine stessa. All'aumentare della risoluzione, cresce anche la densità dei pixel, e quindi ne diminuisce la dimensione.

L'unità di misura della risoluzione spaziale è il *ppi* (pixel per inch) nel caso di visualizzazione su monitor, mentre la risoluzione delle stampanti si misura in *dpi* (dots per inch). Nel caso di stampa tipografica si parla di *lpi* (lines per inch), ed infine, per i filmati tradizionali, la risoluzione si misura in *grains*.

1.2 Interpolazione

Si potrebbe pensare di ottenere risoluzioni sempre maggiori, ampliando indefinitamente il numero di fotorivelatori del sensore (e riducendone proporzionalmente la dimensione); purtroppo, però, c'è un limite inferiore alla dimensione dei campioni, al di sotto del quale il rumore introdotto degrada la qualità del risultato. La miglior soluzione, allora, è di catturare il segnale visivo ad una risoluzione non molto alta, da aumentare poi, algoritmicamente. Di questo si occupa lo Zooming digitale, dunque l'incremento artificiale di risoluzione è anche il principale obiettivo dell'algoritmo DDZ.

Quando guardiamo un oggetto più da vicino, o attraverso una lente di ingrandimento o altro sistema di zooming analogico come teleobiettivi, telecamere ecc., riusciamo a percepire meglio i suoi particolari più piccoli perché il rate di campionamento è maggiore, perciò i livelli di dettaglio che percepiamo sono aumentati.

Nello zooming digitale, invece, la matrice di pixel iniziale è la nostra unica fonte di informazioni; pertanto, anche dopo l'espansione delle dimensioni dell'immagine, il numero dei pixel di cui si conosce il valore rimane invariato, mentre ne servirebbero molti di più per ricoprire la nuova area dello schermo.

Questo scompensamento potrebbe essere affrontato in un modo 'naif': Si potrebbero semplicemente ingrandire tutti i pixel originali, ciascuno con lo stesso fattore di scala, come si vede in figura 1.1.a. In tal modo, però, vengono evidenziati i contrasti fra le aree quadrate dei singoli pixel, ciò è dovuto al fatto che si sta mantenendo la medesima risoluzione dell'originale.

Per evitare questo effetto, la dimensione dei pixel va mantenuta costante, e ne va aumentato il numero, in altre parole, va aumentata la risoluzione dell'immagine; i pixel da aggiungere andranno opportunamente calcolati, a partire da quelli noti, avvalendosi di qualche tecnica di interpolazione.

L'interpolazione è la ricostruzione di una superficie di colore continua a partire da campioni discreti. Se ne conoscono molti metodi, quelli più popolari, e maggiormente argomentati dei software di grafica sono:

• interpolazione

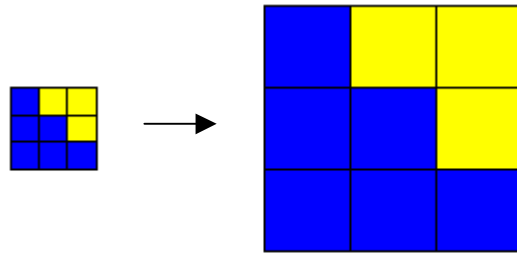


Fig.1.1.a: Aumento della dimensione del pixel: si mettono in evidenza gli errori introdotti dalla digitalizzazione.

1.3 Espansione

Rappresentiamo un'immagine I_1 di dimensioni $w \times h$, con una matrice di pixel con w righe e h colonne. Se vogliamo ingrandirla di un fattore di scala $scale$, il primo passo da realizzare è la semplice espansione delle posizioni dei pixel originari, in una griglia regolare, di dimensioni $\text{floor}(scale * w) \times \text{floor}(scale * h)$. Chiamiamo I_2 l'ingrandimento finale.

Se indichiamo con $I_1(i,j)$ il valore RGB del pixel che sta nella posizione determinata dalla i -esima riga e j -esima colonna di I_1 , e con $I_2(l,k)$ il valore del pixel alla l -esima riga e k -esima colonna di I_2 , l'espansione è una mappatura $E: I_1 \rightarrow I_2$, definita secondo l'equazione:

$$E(I_1(i,j)) = I_2(\text{floor}(i*scale), \text{floor}(j*scale)) \quad (\text{Eq. 1.1})$$

$$i = 1, 2, \dots, w; \quad j = 1, 2, \dots, h$$

La procedura di espansione sarà del tipo:

```
for(int i = 0; i < w; i++)
  for(int j = 0; j < h; j++)
  {
    ii = floor(i*scale);
    jj = floor(j*scale);
    I2[ii][jj] = I1[i][j];
  }
```

La Figura 1.2 mostra una rappresentazione della matrice di pixel così ottenuta, (*‘immagine espansa’* o *‘matrice espansa’* d’ora in poi). I pallini scuri indicano le posizioni in cui sono stati mappati i pixel della matrice originale. Osservando la figura, si vede chiaramente che, dopo aver espanso l’immagine, solo una frazione f , pari a circa $1/scale^2$, del numero dei pixel totali del risultato, sarà immediatamente noto e perfettamente coerente con l’originale; il resto dovrà essere dedotto a partire dal campione di pixel noti.

●	○	●	○	●	○	●	○	●
○	○	○	○	○	○	○	○	○
●	○	●	○	●	○	●	○	●
○	○	○	○	○	○	○	○	○
●	○	●	○	●	○	●	○	●
○	○	○	○	○	○	○	○	○
●	○	●	○	●	○	●	○	●
○	○	○	○	○	○	○	○	○
●	○	●	○	●	○	●	○	●

Fig. 1.2: Immagine Espansa con fattore di scala 2.
I pallini neri indicano le posizioni in cui il pixel è noto.

1.4 Lo stato dell'arte

Tra le tecniche di zooming note, vi sono tecniche classiche, come:

- Replica dei pixel (Nearest Neighbor);
- Interpolazione Bilineare;
- Interpolazione Bicubica;

E tecniche di recente pubblicazione, come:

- L.A.Z.A. (BGS), sviluppata dall'università di Catania;
- Zooming di ReST, sviluppata dall'AST-Lab, STMicroelectronics;
- Adaptive Zooming di Li Hong;

-Replica e Interpolazione Bilineare-

La replica dei pixel, è la tecnica più immediata fra tutte, ma produce l'approssimazione meno accurata, essa consiste nel replicare ciascun pixel originale e copiarlo in tutte le posizioni dell'area quadrata circostante, in maniera tale da riempire tutte le aree intermedie tra i pixel noti. In pratica, ogni pixel incognito assume lo stesso valore del pixel ad esso più vicino.

L'interpolazione Bilineare è l'estensione 2D dell'interpolazione lineare, che si può sintetizzare così: preso il valore iniziale, i , e il valore finale, f (entrambi noti), della parte incognita del segnale, tutti i punti intermedi variano linearmente da i a f . Replica e interpolazione lineare sono mostrate in figura 1.3.a. Per passare al campo bidimensionale, basta applicare successivamente lo stesso procedimento lungo entrambe le direzioni degli assi cartesiani, la figura 1.3.b ne mostra un esempio.

In pratica l'interpolazione bilineare assegna ad ogni pixel un valore ottenuto mediando sugli *scale* x *scale* valori dei pixel più vicini ad esso.

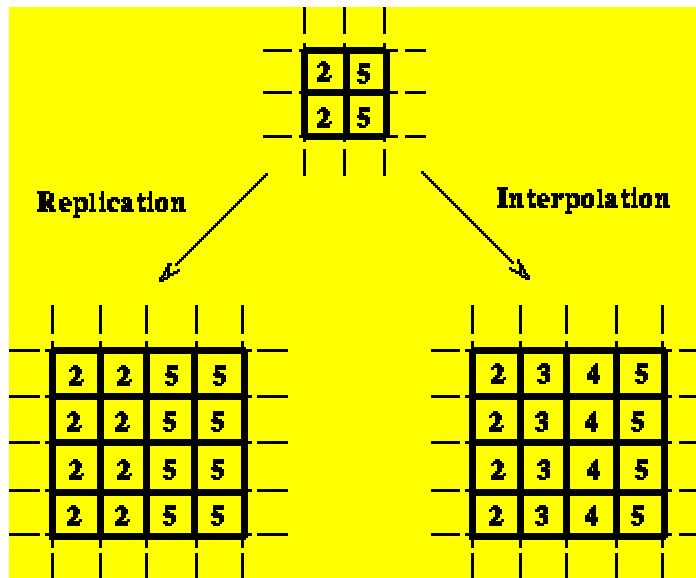


Fig. 1.3.a: Un esempio di Replica dei pixel, e di Interpolazione Lineare.

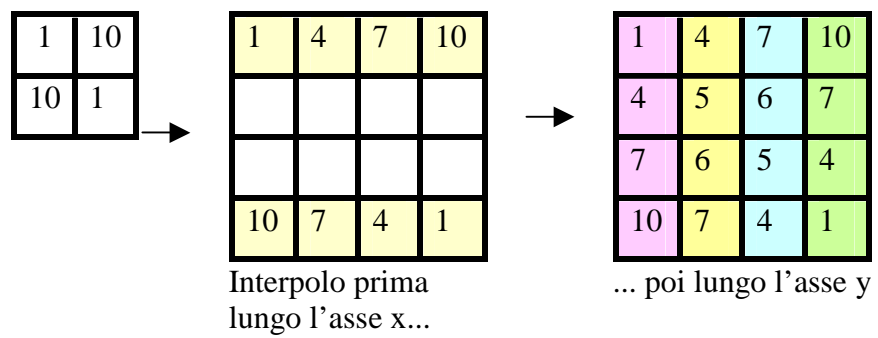


Fig. 1.3.b: Un esempio di Interpolazione Bilineare.

-Interpolazione Bicubica-

L'approccio standard è l'interpolazione bicubica, essa assegna il colore ad un pixel nell'immagine risultato, calcolando una media di $scale^2 \times scale^2$ (sedici, nel caso di raddoppio delle dimensioni) pixels situati nell'intorno del pixel corrispondente, nell'immagine d'origine.

Esistono due metodi comuni di interpolazione bicubica: cubic B-Spline, ed una funzione interpolatoria cubica. In questa sede, si descriverà la versione basata sulle B-Splines cubiche, in modo da rispecchiare l'analoga scelta fatta nell'implementazione dell'interpolazione bicubica, inserita nell'applicazione DDZ, ai fini di validazione sperimentale della tecnica di zooming proposta.

Sia (i',j') il punto dell'immagine destinazione, di cui si vuole determinare il colore. Siano (i,j) le più vicine coordinate intere, nell'immagine originale, e dx e dy , le loro rispettive differenze.

L'equazione 1.2 mostra la formula del valore interpolato, applicata al grado di grigio (se l'immagine è a toni di grigio) o, in caso contrario, al singolo canale di colore R, G, o B. Per semplicità si è posto $scale = 2$, di conseguenza, le sommatorie si estendono nell'intorno 4×4 del pixel (i,j) .

$$F(i', j') = \sum_{m=-1}^2 \sum_{n=-1}^2 F(i + m, j + n) R(m - dx) R(dy - n)$$

dove $R(x)$ è la seguente funzione cubica:

$$R(x) = \frac{1}{6} [P(x + 2)^3 - 4 P(x + 1)^3 + 6 P(x)^3 - 4 P(x - 1)^3]$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

- Locally Adaptive Zooming Algorithm (LAZA)-

L'algoritmo LAZA (o BGS), compie un'interpolazione pesata regolata dal gradiente; a differenza degli altri algoritmi adattivi, non richiede una precomputazione del gradiente, e si svolge in quattro fasi:

1. Espansione. Cfr. paragrafo 1.3.
2. Per ogni pixel x nella matrice espansa, di coordinate entrambe dispari, si distinguono i seguenti casi: (per semplicità, $scale = 2$)
 - a. *Uniformità*: se il range dei quattro pixel noti attorno ad x , è abbastanza contenuto, gli si assegna la media dei quattro valori.
 - b. *Arco diagonale*: si assegna ad x la media dei valori vertici.
 - c. *Arco verticale*: si assegnano al pixel in alto e al pixel in basso ad x , i rispettivi valori medi dei loro pixel laterali, x indefinito.
 - d. *Arco orizzontale*: i pixel laterali assumono il valore medio dei loro vicini superiore e inferiore, x indefinito.
3. Per ogni pixel lasciato indefinito al passo precedente, e con almeno una coordinata dispari :
 - a. Se i due pixel ad esso vicini, lungo la coordinata dispari in considerazione, sono anch'essi indefiniti, allora si verifica se i valori dei due pixel ad esso vicini lungo l'altra coordinata sono sufficientemente simili, in tal caso, se ne assegna la media al pixel centrale. Altrimenti, lo si lascia indefinito.
 - b. Nel caso opposto, si verifica la presenza eventuale di un arco orizzontale o verticale, nei quali casi si assegna al pixel centrale, il valore medio dei due pixel appartenenti all'arco, altrimenti lo si lascia indefinito.

4. Inizialmente si suddivide il range dei valori di colore (tipicamente 0 - 255), in una scala più grossolana, di $256/q$ gruppi costituiti da q valori consecutivi, con q intero. L'algoritmo trasforma i pixel originari circostanti il pixel incognito, in almeno uno e al più quattro gruppi. Il valore del pixel incognito, è calcolato come la media dei valori rappresentanti di ciascun gruppo (tipicamente il suo valore medio) assegnato attorno ad esso.

-ReST-

L'algoritmo ReST prende in considerazione le discontinuità tra i pixel dell'immagine d'origine. Senza ledere la generalità, si discuterà il caso particolare di fattore di scala 2. Per ciascun pixel originale, l'algoritmo procede secondo 3 passi:

1. Si investiga sulla eventuale presenza, nel contorno 3x3 del pixel corrente, di un arco verticale, orizzontale, diagonale-dx, diagonale-sx.
2. Se si è trovato un arco, i quattro pixels corrispondenti al corrente, nell'immagine ingrandita, sono calcolati di conseguenza, facendo la media dei massimi e minimi relativi direzionali.
3. I quattro pixel, così generati dal pixel dell'immagine a bassa risoluzione, vengono memorizzati nelle posizioni corrispondenti dell'immagine ad alta risoluzione.

-Algoritmo Adattivo-

Si tratta di un algoritmo specifico per fattore di scala 2, e lavora in tre fasi:

1. Divide l'immagine, in blocchi non disgiunti di 4x4 pixel.
2. per ciascun blocco, calcola la media dei sedici valori, e se i pixel del blocco hanno valori tutti ad essa vicini, lo si classifica come 'smooth'. Altrimenti, si marca ogni pixel con '+' o '-' a seconda che sia minore o maggiore alla media, se c'è una sola componente connessa di pixel '+', allora è un blocco 'edge', altrimenti è 'texture'.
3. Ogni tipo di blocco, si interpola diversamente, con filtri più o meno lisci, se si tratta di blocchi 'smooth' o 'texture', mentre per blocchi 'edge', si interpola lungo la direzione prevalente dell'arco.

Capitolo 2 : Triangolazione Dipendente dai Dati

La DDT è stata introdotta da Nira Dyn, David Levin e Samuel Rippa, nel 1990 con la pubblicazione del loro “Data Dependent Triangulations for Piecewise Linear Interpolation” sull’IMA Journal of Numerical Analysis.

2.1 La Triangolazione – Definizione del problema

Informalmente, si tratta di partizionare il piano dell’immagine in triangoli con vertici nei punti corrispondenti a pixel noti (scegliendo tra le possibili partizioni, quella ottimale rispetto ad un certo criterio), e ricostruire il colore dei pixel interni di ogni triangolo mediante interpolazione lineare sui vertici.

Più formalmente, iniziamo dal problema generale della triangolazione:

Data una regione poligonale Ω del piano, ed un insieme $V = \{(x_i, y_i)\}_{i=1,2,\dots,N}$ di punti del piano, non tutti collineari, ad essa interni, sia f una funzione binaria a valori reali, ivi definita ma incognita

$$f(x, y): \Omega \Rightarrow \mathbb{R}$$

Siano inoltre noti i valori assunti dalla f negli N punti dati.

$$f(x_i, y_i) = F_i \quad i=1,2,\dots,N$$

Una Triangolazione su Ω è un insieme $T = \{T_i\}_{i=1,2,\dots,q}$ di triangoli non degeneri che soddisfa le seguenti condizioni:

1. V è l’insieme di tutti e soli i vertici dei triangoli di T .
2. Per ogni triangolo di T , nessuno dei suoi lati passa per alcun punto $\in V$, esclusi gli estremi.

3. T è una partizione di Ω , cioè:

$$\Omega = \cup_{\forall T_i \in T} (T_i)$$

$$T_i \cap T_j = \emptyset \quad \forall T_i, T_j \in T : i \neq j$$

Ogni triangolazione su Ω possiede lo stesso numero di triangoli, q , e il medesimo numero di archi, r :

$$q = 2(N - 1) - N_b, \quad r = 3(N - 1) - N_b$$

dove N_b è il numero di vertici giacenti sulla frontiera di Ω .

Nel caso dello zooming, Ω è il rettangolo rappresentato dall'area della matrice espansa (fig. 1.1), la funzione f (che essendo digitale è una funzione discreta definita su tutti i pixel della stessa matrice) può essere ad esempio l'intensità nelle immagini a toni di grigio, o la luminanza o la terna di valori dei canali R,G e B in quelle a colori, V è l'insieme delle posizioni dei pixel noti nella matrice, e gli F_i sono i loro rispettivi valori.

I dati a nostra conoscenza sono, quindi, le triple:

$$(x_i, y_i, F_i) \quad : \quad \text{Pixels}[x_i][y_i] = F_i \quad \text{per } i=1,2,\dots,N$$

Questo ci permette di lavorare nello Spazio 3D ponendo l'area dell'immagine Ω , giacente sul piano x-y, e come terza coordinata, la funzione f applicata ai punti del piano.

Una volta scelta una delle possibili triangolazioni, T , per colmare le aree interne ad ogni triangolo T_i di T si calcolerà l'unico polinomio lineare in 3D che interpola f nei vertici di T_i , cioè l'equazione dell'unico piano passante per i tre punti dello spazio (x_1, y_1, F_1) , (x_2, y_2, F_2) , (x_3, y_3, F_3) .

L'interpolatore sarà quindi del tipo:

$$P_i(x,y) = a_i x + b_i y + c_i \quad \Big| \quad T_i \quad (\text{Eq. 2.1})$$

La funzione f geometricamente è una superficie tridimensionale continua, se partizioniamo il suo dominio, che è la sua proiezione sul piano $\langle x,y \rangle$, ottenendone una triangolazione, possiamo considerare f come l'unione di porzioni triangolari disgiunte di superfici S_i , con i lati, fra le coppie di triangoli contigui, in comune.

L'approssimazione di f che si ottiene col metodo qui descritto, è una superficie tridimensionale continua, formata dall'unione delle singole approssimazioni delle S_i con piani P_i , definiti nei corrispondenti domini delle S_i .

Questo genere di superficie nello spazio 3D si chiama 'PLIS' (Piecewise Linear Interpolating Surface).

La questione fondamentale è riuscire a scegliere la migliore triangolazione possibile, ossia quella ottimale, che permetta di approssimare la f con una PLIS il più possibile vicina alla superficie 3D ideale che si otterrebbe con una maggiore risoluzione nell'acquisizione del segnale.

Per poter parlare di Triangolazione Ottimale si deve prima scegliere un criterio di ottimalità.

Prescindendo dalla applicazione al restauro di immagini, consideriamo il problema generale, in cui i punti noti sono situati in posizioni non necessariamente regolari del piano, e l'area Ω è un poligono generico.

2.2 Triangolazione di Delaunay

I criteri di ottimalità precedenti alla DDT, si basavano sulle proprietà geometriche bidimensionali della distribuzione dei vertici $V=\{(x_i,y_i)\}_{i=1,2,\dots,N}$; quindi la triangolazione da essi prodotta risultava dipendente soltanto dall'insieme dei punti V , e invariante rispetto all'insieme dei dati $\{F_i\}_{i=1,2,\dots,N}$.

I due criteri più diffusi erano: la triangolazione min-max, che massimizza il minimo angolo presente nell'insieme dei triangoli, e la triangolazione max-min, che minimizza il massimo angolo;

L'idea fondamentale era che i triangoli dovessero essere il più possibile equiangoli, il che, per qualche particolare applicazione, può anche essere vero, ma non lo è in generale, come Samuel Rippa ha dimostrato nel suo articolo “Long, thin triangles can be good for linear interpolation”, apparso nel SIAM Journal of Numerical analysis nel 1992.

La Triangolazione di Delaunay fu introdotta nel 1934 da Delaunay. Essa si basa sul criterio min-max, soddisfacendo la condizione che, se la diagonale di un quadrilatero viene sostituita dalla sua opposta, il minimo degli angoli interni al quadrilatero non aumenta. Un'utile proprietà della Triangolazione di Delaunay è la sua **unicità**, che la rende particolarmente adatta a scopi di codifica, giacché per la decodifica basta conoscere la posizione degli N punti.

Definizioni preliminari:

- Il circocerchio di un triangolo è l'unico cerchio del piano che passa per tutti e tre i suoi vertici.

- Un triangolo è detto *triangolo di Delaunay*, se e solo se il suo circocerkio non contiene alcun vertice di V .
- Si chiama triangolazione di Delaunay ogni triangolazione i cui triangoli sono tutti *triangoli di Delaunay*.

La costruzione di una triangolazione di Delaunay, in genere, viene definita mediante il suo problema duale, ossia il **Diagramma di Voronoi**:

Dato un insieme V di N punti del piano, per ogni $p_i \in V$, si vuole determinare il luogo dei punti del piano che sono vicini a p_i più di quanto lo sono rispetto agli altri punti di V , che si chiama Poligono di Voronoi associato a p_i , e si indica con $V(i)$.

Dati $p_i, p_j \in V$, distinti, il luogo dei punti del piano più vicini a p_i che a p_j è il semipiano contenente p_i , definito dall'asse del segmento $p_i p_j$, lo si indichi con $H(p_i, p_j)$. Allora:

$$V(i) = \bigcap_{i \neq j} H(p_i, p_j)$$

La rete convessa formata dalla partizione del piano negli N poligoni di Voronoi associati ai punti di V , è il Diagramma di Voronoi per V , e si indica con $Vor(V)$.

La soluzione proposta da Delaunay è un metodo divide-et-conquer, ed è così schematizzato:

- Si partizioni V in due sottoinsiemi della stessa dimensione, V_1 e V_2 dividendoli in base alla coordinata x .
- Si costruiscano $Vor(V_1)$ e $Vor(V_2)$ ricorsivamente.
- Si fondano $Vor(V_1)$ e $Vor(V_2)$ ottenendo $Vor(V)$.

L'algoritmo è $\Theta(N \log N)$.

La dualità dei due problemi è stata dimostrata da Delaunay stesso, ma in questa sede basterà osservare la figura 2.1 per averne un'idea intuitiva:

In figura 2.1.a è rappresentato un diagramma di Voronoi, per ottenere la corrispondente Triangolazione di Delaunay su V , procediamo su di esso nel seguente modo: per ciascun segmento s , e i due punti p_1 e p_2 che esso separa, cancelliamo s e sostituiamolo con il segmento s_I che unisce p_1 e p_2 . Il risultato è mostrato in figura 2.1.b ed è proprio la Triangolazione di Delaunay su V .

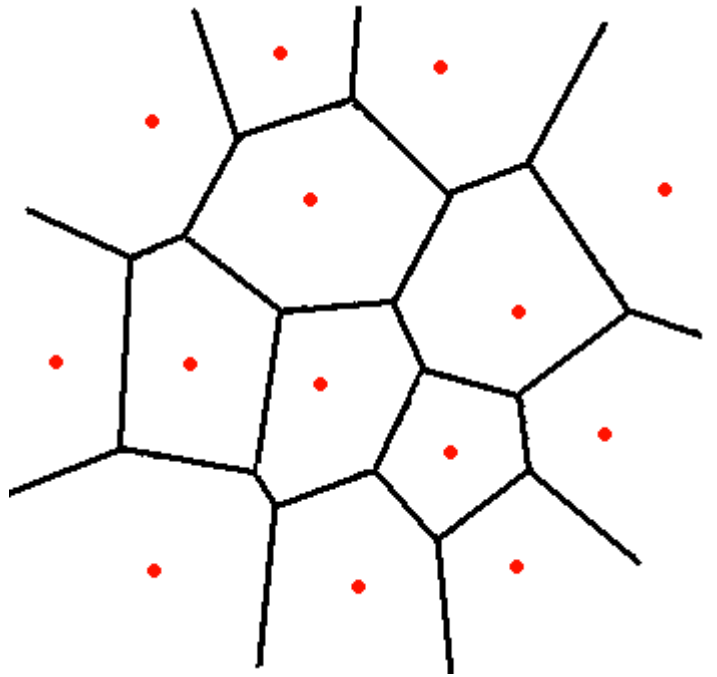


Fig 2.1.a: un Diagramma di Voronoi

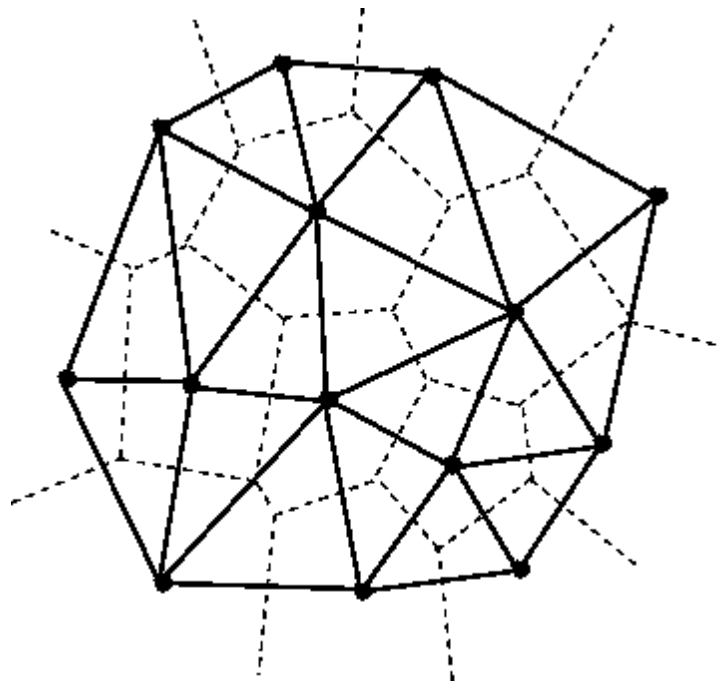


Fig 2.1.b: La Triangolazione di Delaunay corrispondente al Diagramma di Voronoi di figura 2.1.a

2.3 Data Dependent Triangulation

La DDT, a differenza dei metodi tradizionali, sfrutta produttivamente anche l'informazione fornita dai dati F_1, F_2, \dots, F_N per minimizzare l'impatto visivo degli inevitabili errori di ricostruzione. Seppur non riducendone drasticamente la quantità, minimizza l'introduzione di smoothing e di artefatti nelle alte frequenze.

Il nome "Data Dependent" deriva proprio dal fatto che la scelta della triangolazione viene effettuata in conformità a criteri di ottimalità dipendenti non solo dalle posizioni (x_i, y_i) ma da tutti e tre gli elementi (x_i, y_i, F_i) a noi noti sugli N pixel, essi infatti prendono in considerazione le proprietà geometriche tridimensionali dei piani interpolatori anziché soltanto quelle 2D delle loro proiezioni sul piano $\langle x, y \rangle$.

E' stato dimostrato da Nira Dyn, David Levin e Samuel Rippa, che la Data Dependent Triangulation offre una migliore approssimazione della f in confronto alla triangolazione di Delaunay (cfr. paragrafo 2.2).

Il motivo della grande utilità della dipendenza dai dati sta nel fatto che, considerando i valori F_1, F_2, \dots, F_N come campioni di una superficie incognita $f(x, y)$, in molti casi, siamo in grado di dedurre ulteriori informazioni sulle caratteristiche di $f(x, y)$. Ad esempio si può dedurre la monotonicità e la convessità, oppure qualche proprietà variazionale o l'entità e l'orientamento dei dislivelli, ecc.

La Triangolazione ottimale rispetto ad un dato criterio esiste sempre, perché il numero totale di possibili triangolazioni è finito, ma trovarla può essere difficile nella pratica.

Molto utile per gli scopi pratici è la definizione di *Triangolazione Localmente ottimale*. Sia T una triangolazione su Ω , sia e un arco interno di T , e sia Q il quadrilatero formato dai due triangoli di T che condividono e .

L'arco e è un *Arco Localmente Ottimale secondo un criterio di ottimalità* se, scambiando e con la diagonale opposta di Q , e lasciando invariati tutti gli altri archi della triangolazione, si ottiene una triangolazione peggiore secondo il criterio scelto. La triangolazione T si dice *Localmente Ottimale*, se tutti gli archi di cui è composta sono *Localmente Ottimali*.

La ricerca della triangolazione localmente ottimale si può effettuare facilmente con il seguente algoritmo ideato da C.L.Lawson:

Local Optimization Procedure (LOP)

1) Costruisci una triangolazione iniziale T_0 e assegna

$$T \leftarrow T_0$$

2) Se T è Localmente Ottimale, esci

3) Sia e un arco interno di T non localmente ottimale, e sia Q il quadrilatero strettamente convesso formato dai due triangoli che condividono e ,

3.a) Scambia e con la diagonale opposta di Q , e_1 ,

$$\text{ottenendo: } T_1 = T \cup \{e_1\} \setminus \{e\}$$

3.b) Assegna $T \leftarrow T_1$ e torna al passo 2

L'algoritmo converge (e si è visto che lo fa in poche iterazioni) ad una triangolazione T_h localmente ottimale, perché ad ogni scambio il costo di un arco diminuisce e di conseguenza decresce il costo totale, che non può scendere all'infinito e arriverà ad un valore minimo da cui non si sposterà più.

La triangolazione localmente ottimale ottenuta dal LOP può dipendere dallo specifico ordine in cui gli archi sono scambiati; tuttavia, l'ottimalità permane.

2.4 Criteri di Ottimalità.

La scelta del criterio data-dependent, deve essere influenzata da due concetti fondamentali: la natura dei dati rappresentati da $f(x,y)$, e lo scopo finale della loro approssimazione. Riguardo alla natura dei dati, a seconda del fenomeno che essi descrivono, un criterio sarà più adeguato di altri.

Nel caso delle immagini, i dati sono le intensità dei pixel, e la superficie tende ad avere due tipi di comportamento nel suo dominio: può avere delle sottoaree in cui è prevalentemente liscia e omogenea, mentre le alte frequenze tendono ad avere una direzione preferita, vale a dire, la derivata seconda lungo una direzione è maggiore rispetto alle altre.

In quest'ultima situazione, ad esempio, un buon criterio permetterà la costruzione di triangoli stretti e lunghi che presentino il lato maggiore orientato nella direzione di minore curvatura di $f(x,y)$.

D'altro canto, la disposizione regolare dei punti di V rende poco appropriate le considerazioni sulla prossimità, o sulle altre caratteristiche dei punti stessi;

L'altro aspetto da non sottovalutare, è l'utilità finale. Ad esempio, per scopi di codifica e decodifica del segnale, il più indicato criterio è quello alla base della triangolazione di Delaunay (cfr. paragrafo 2.2). Al contrario, nello zooming, la natura grafica dell'applicazione comporta la necessità di considerazioni sull'aspetto finale dell'output, come l'esigenza di minimizzare gli artefatti introdotti, che esulano dal problema dell'accuratezza dell'approssimazione.

Supponiamo di poter accedere, come verifica esterna della bontà della soluzione trovata, al valore della f in ogni punto del suo dominio, (non inseriamo questo dato nel nostro problema, altrimenti, senza incognite, esso non esisterebbe!).

Potremmo definire ottimale la triangolazione che produce un'approssimazione della f numericamente più accurata punto per punto. Questa è sicuramente una buona idea dal punto di vista teorico, considerando il problema solo come la ricerca dell'approssimazione analitico-geometrica di una funzione binaria discreta, nota in un sottoinsieme proprio di punti del dominio, a due a due non contigui.

Tuttavia, nell'applicare l'idea della triangolazione alla ricostruzione di immagini, non possiamo concentrare la nostra attenzione soltanto sull'aspetto matematico della questione, poiché non è detto che l'approssimazione dei valori dei pixel mancanti, che, tra tutte, introduce la quantità minima di errore numerico, poi dia luogo ad un risultato visivamente meno pieno di artefatti ed effetti sgradevoli, o più simile, ad occhio, all'immagine di partenza.

2.5 Funzioni costo

Per determinare una misura di ottimalità delle triangolazioni, bisogna poter associare un ‘ livello di bontà ’ alle varie triangolazioni definibili su Ω , deducibile in maniera coerente da quello dei loro archi costituenti. (Con ‘ livello di bontà ’ si intende una misura di quanto ci si avvicini alla Locale Ottimalità rispetto al criterio fissato).

Indicando con $costo(T)$ l'inverso della misura di bontà associata alla triangolazione T , una triangolazione T è da preferire ad una triangolazione T^l , definita sullo stesso dominio, se e solo se : $costo(T) < costo(T^l)$.

Per ogni triangolazione T , si definisce una funzione reale $costo$ sul dominio dei suoi archi, e si impone che il costo complessivo della stessa vada calcolato dai valori $costo$ degli archi, mediante una norma che preservi la proprietà che:

$$costo(e) < costo(e^l) \Leftrightarrow costo(T) < costo(T \cup \{e^l\} \setminus \{e\})$$
$$\forall \text{ arco } e \in T, \text{ e la sua diagonale opposta } e^l$$

Tra le norme proposte in letteratura, si è scelta la norma L_1 :

$$costo(T) = \sum_{e \in T} | costo(e) | \quad (\text{Eq. 2.2})$$

La *funzione-costo* deve essere definita in maniera che sia efficientemente computabile, non solo perché il numero degli archi può essere molto alto, ma anche perché i costi dei nuovi archi andranno calcolati run-time man mano che la triangolazione si evolve.

2.5.1 Funzioni NC1

Una vasta categoria di superfici $f(x,y)$, presenta una certa ‘uniformità’ sull’insieme dei dati, cioè non oscilla troppo tra i punti di V .

Dyn, Levin e Rippa, hanno proposto una classe di funzioni-costo denominata NC1 (Nearly C^1), particolarmente appropriata per questo genere di superfici.

Esse, infatti, misurano caratteristiche proprie delle interfacce tra due triangoli adiacenti. Sia e un arco nella triangolazione, siano T_1 e T_2 i due triangoli adiacenti fra loro tramite e , e siano P_1 e P_2 i loro polinomi interpolatori, cioè, dall’equazione 2.1 :

$$P_1(x,y) = a_1 x + b_1 y + c_1 \Big|_{T_1}$$

$$P_2(x,y) = a_2 x + b_2 y + c_2 \Big|_{T_2}$$

Le funzioni NC1 sono le seguenti:

- La funzione costo ABN (Angle Between Normals), sull’arco $e \in T$, è il coseno dell’angolo tra le normali ai piani interpolatori P_1 e P_2 .
- Il costo JND (Jump in Normal Derivatives) di e , ritorna il valore:

$$| n_x(a_1 - a_2) + n_y(b_1 - b_2) | ,$$

dove (n_x, n_y) è un vettore unitario giacente sul piano $\langle x, y \rangle$, ortogonale alla proiezione dell’arco e .

- La DLP (Deviation from Linear Polinomial) applicata ad e , dà il valore $|h|$, dove h è il vettore:

$$h = \begin{bmatrix} |P_1(x_1, y_1) - F_1| \\ |P_2(x_2, y_2) - F_2| \end{bmatrix}$$

- La DP (Distance from Planes) misura la distanza tra ciascuno dei due piani, P_1 e P_2 , e il vertice opposto nel quadrilatero formato da T_1 e T_2 . Essa restituisce $|g|$, dove g è il vettore:

$$g = \begin{bmatrix} \text{dist}(P_1, w_2) \\ \text{dist}(P_2, w_1) \end{bmatrix},$$

$$\text{e } \text{dist}(P_i, w_j) = \frac{|P_i(x_j, y_j) - F_j|}{(a_i^2 + b_i^2 + 1)^{1/2}}$$

Yu, Morse e Sederberg, hanno implementato e testato ricostruzioni d'immagini basate su ciascuna di queste quattro funzioni-costo, e la migliore è risultata DP. Purtroppo, però, anch' essa presenta una certa sensibilità al variare della gamma dei valori d'intensità delle immagini, che dipende da proprietà specifiche dell'hardware e del software che si usa.

La funzione-costo scelta in DDZ gode, invece, della proprietà d'indipendenza dal range delle intensità.

Si rimanda al paragrafo 3.3 per una descrizione dettagliata della suddetta funzione.

Capitolo 3 : Data Driven Zooming (DDZ)

3.1 Introduzione

Per questa tesi è stata realizzata un'implementazione software della tecnica di Zooming mediante Data Dependent Triangulation (algoritmo DDZ, presentato in questo capitolo), nonché una serie di altre funzionalità ad essa correlate. L'applicazione si chiama DDZ, il file eseguibile è allegato alla tesi in formato compresso, "ddz.zip".

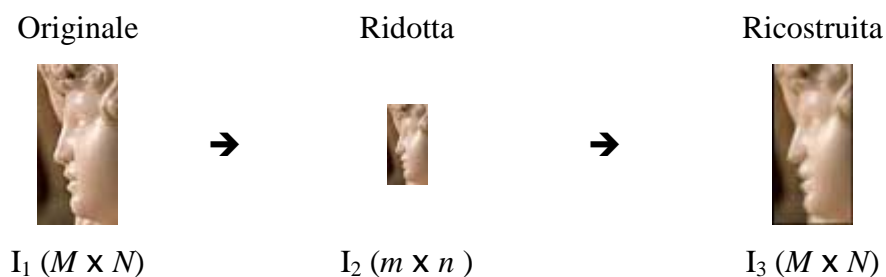
Le principali funzionalità incluse nel programma sono:

- Implementazione dell'algoritmo DDZ, in quattro versioni:
 - Implementazione dell'algoritmo di Ricostruzione di immagini mediante Data Dependent Triangulation di Yu, Morse, Sederberg;
 - Ottimizzazione originale dello stesso;
 - Implementazione dell'algoritmo di Interpolazione di immagini mediante Pixel Level Data Dependent Triangulation ideato da Su e Willis;
 - Il nuovo algoritmo Data Driven Zooming.
- Implementazione del metodo di Zooming attualmente più usato in commercio, l'interpolazione bicubica.
- Implementazione dell'algoritmo di Zooming più semplice e immediato: la replica.
- Software dimostrativo che offre la possibilità di testare, su qualsiasi immagine digitale, tutti gli algoritmi implementati, e di confrontarne i risultati.

- Opzione “Sottocampionamento → Reingrandimento”, che permette di confrontare l'immagine originale con la sua approssimazione, ottenuta applicando uno dei metodi di Zooming all'immagine rimpicciolita. (cfr. schema 3.1).
- Calcolo del PSNR (funzione di misura oggettiva dell'apporto di errori) sulle immagini ricostruite con la suddetta opzione.
- Gestione interattiva delle iterazioni dell' algoritmo DDZ, con visualizzazione e PSNR dei risultati intermedi.
- Visualizzazione di una serie di grafici che mostrano i risultati statistici del testing della tecnica presentata, a confronto con gli altri algoritmi all'avanguardia nel campo (cfr. 1.4), svolto su un campione di immagini di vario genere.

L'opzione “Sottocampionamento → Reingrandimento” consiste nel sottocampionare un'immagine nota I_1 rimpicciolandola di un fattore di scala $scale < 1$, ottenendo l'immagine ridotta I_2 , e applicare a quest'ultima uno zooming di scala $1/scale$, ottenendo I_3 .

Schema 3.1



dove $M = \text{floor}(m \cdot \text{scale})$, $N = \text{floor}(n \cdot \text{scale})$, nell'esempio $scale = 2$

DDZ è un algoritmo iterativo, costruito sulla base della Local Optimization Procedure di Lawson per la ricerca della triangolazione localmente ottimale (cfr. 2.3); adattato nel modo di seguito descritto:

Si parte da una particolare triangolazione iniziale T_0 (cfr. 3.4.1) e, finché non si raggiunge una triangolazione localmente ottimale, si eseguono ulteriori iterazioni che effettuano tutti i possibili scambi di diagonale in cui il nuovo arco abbia un *costo* inferiore al vecchio.

In realtà l'utente ha la possibilità di guidare personalmente l'esecuzione del programma, avviando ciascun'iterazione autonomamente, e visualizzandone sullo schermo il risultato intermedio. In questo modo, egli può interrompere la computazione in qualunque momento ne ritenga l'esito soddisfacente per l'uso finale.

Dopo uno studio sperimentale del comportamento del programma (svolto parallelamente al testing del cap.4), si è visto che la triangolazione va stabilizzandosi sempre più, fino a convergere alla triangolazione localmente ottimale rispetto al criterio scelto (cfr. funzione costo al paragrafo 3.3), in altre parole, ogni eventuale ulteriore iterazione non produce alcun cambiamento. (cfr. 4.2.1)

3.2 Strutture Dati

La Struttura Dati su cui DDZ lavora è simile ad un grafo $G=(V,E)$ in cui: I vertici sono i pixel di cui è noto il valore RGB, disposti a griglia nell'immagine espansa, e sono oggetti di classe *puntocol*, con campi per le coordinate e per i valori RGB.

Gli archi sono oggetti di classe *arco*, che mantiene informazioni sugli estremi e sul costo (ottenuto applicandovi la *funzione-costo*), ed ha un campo booleano flag che distingue gli archi localmente ottimali.

Data la definizione di triangolazione (cfr. 2.1), gli archi sono organizzati in triangoli contigui, ogni arco presente nel grafo appartiene, come lato, a due triangoli fra loro adiacenti; ciascuno di essi non può essere considerato né semplicemente arco diretto, né indiretto, in realtà è una **coppia di archi diretti** che collegano gli stessi due vertici, ma non necessariamente di verso opposto l'uno dall'altro, perché il verso che ogni arco assume in uno dei due triangoli cui appartiene come lato, è indipendente dal verso che assume nell'altro triangolo. Pertanto, ciascun arco è istanziato con due oggetti diversi, uno per *triangolo*, entrambi diretti, ma con versi mutuamente indipendenti. G è, quindi, come una coppia di grafi con stesso V , stessa collocazione e costo degli archi di E , ma versi indipendenti. Il grafo non è implementato

La classe *arco* ha i seguenti campi :

- I due vertici u, v (l'*arco* è diretto da u a v)
- Il suo costo (risultante dall'applicazione della *funzione-costo*)
- Un flag che ne indica l'eventuale condizione di non-ottimalità locale

I vertici, a loro volta sono oggetti corredati di:

- Coordinate x, y per localizzarli nell'immagine
- Valori RGB del pixel corrispondente (noti per costruzione)

3.3 Funzione Costo

La *funzione-costo* rappresenta, per quanto precedentemente detto, il criterio di ottimalità (cfr. 2.4 e 2.5). DDZ utilizza una *funzione-costo* ideata da Xiaohua Yu, Bryan S. Morse e Thomas W. Sederberg; essi hanno notato che ricostruzioni i cui contorni sono più lisci tendono a produrre artefatti meno evidenti, e così si sono concentrati sulle curve di livello delle PLIS relative alle diverse triangolazioni, definendo una *funzione-costo* che indica come migliori le PLIS che presentano minori discrepanze tra le curve di livello delle coppie di triangoli adiacenti. Più precisamente, dato un arco e e i due triangoli che lo condividono come lato, T_1 e T_2 , di piani interpolatori P_1 e P_2 rispettivamente,

$$\text{cost}(e) = |\nabla P_1| * |\nabla P_2| * (1 - \cos\theta) \quad (\text{Eq. 3.1})$$

dove: θ è l'angolo tra le normali alle curve di livello di P_1 e P_2 , ciascun $\nabla P_i = (a_i, b_i)$ è il gradiente della superficie piana P_i , a_i e b_i ne sono i coefficienti (cfr. eq. 2.2), quindi $|\nabla P_i| = (a_i^2 + b_i^2)^{1/2}$.

Si ha:

$$\text{cost}(e) = |\nabla P_1| * |\nabla P_2| - \nabla P_1 \cdot \nabla P_2 \quad (\text{Eq. 3.2})$$

$$\Rightarrow \text{cost}(e) = (a_1^2 + b_1^2)^{1/2} * (a_2^2 + b_2^2)^{1/2} - (a_1 * a_2 + b_1 * b_2) \quad (\text{Eq. 3.3})$$

Dall'equazione 2.2 e dalla 3.3, ricaviamo il *costo* dell'intera triangolazione T :

$$\text{costo}(T) = \sum_{\substack{t, t' \in T, \text{adiacenti tra loro,} \\ \text{di interpolatori } P, P'}} \left| (a^2 + b^2)^{1/2} (a'^2 + b'^2)^{1/2} - (aa' + bb') \right|$$

3.4 Il corpo dell'algoritmo

L'algoritmo di Zooming vero e proprio (vale a dire solo il codice che si occupa di ingrandire I_2 restituendo I_3 , cfr. schema 3.1) si articola in due fasi: un preprocessing e un procedimento iterativo.

3.4.1 Preprocessing

Anzitutto, si espande l'immagine piccola I_2 , del fattore di scala *scale*, e si mostra il risultato parziale in una nuova finestra sullo schermo.

Subito dopo, la procedura di **Inizializzazione** si occupa della creazione di un Array Nt di oggetti che rappresentano i triangoli di una Triangolazione iniziale T_0 .

Questa fase prevede la creazione degli archi e dei triangoli che essi formano, inoltre vengono assegnati, per la prima volta, i costi degli archi, ed in ciascun oggetto *triangolo* costruito, vengono registrati i tre polinomi che interpolano l'intensità di rosso, verde e blu, dei vertici (in pratica il calcolo dei relativi coefficienti, cfr. equazione 2.1), per non doverli calcolare ogni volta che si valuta il costo di un suo arco; in questo modo, invece, dopo l'inizializzazione, l'operazione di assegnazione di polinomi interpolatori avverrà solo all'atto della creazione dei triangoli, quando, a causa di uno scambio di diagonali, due vecchi triangoli scompaiono e sono sostituiti da altri due (vedi figura 3.3).

A ciascun triangolo è, inoltre, assegnato un flag booleano utile all'ottimizzazione dell'algoritmo, *opt*, inizialmente posto a zero ovunque (cfr. ottimizzazione).

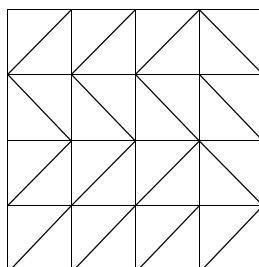


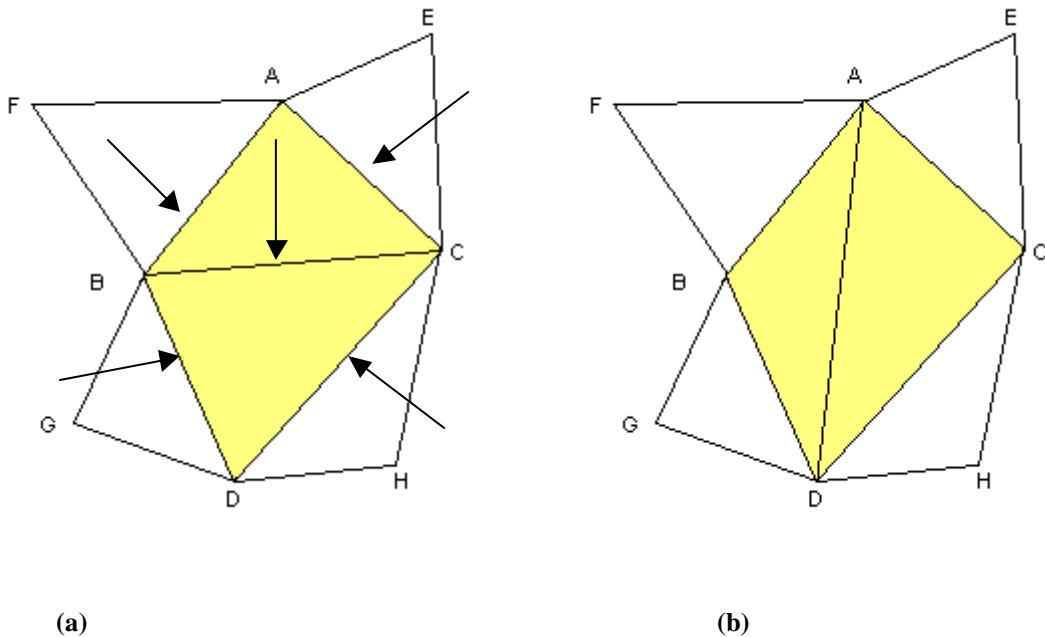
Fig. 3.1: una possibile T_0

La triangolazione iniziale T_0 è semplice: tutti i quadrati individuati in modo ovvio dalla griglia dei vertici (vedi figura 3.1), sono divisi in due triangoli da una delle due possibili diagonali, scegliendo ovviamente quella con *costo* minore.

Per ogni quadrato si calcolano entrambi i costi delle possibili diagonali, e viene creata la coppia di triangoli corrispondente alla diagonale meno costosa, inizializzata in tutti i suoi campi, e registrata all'interno dell'array Nt .

In realtà, fatta eccezione per gli archi che giacciono sulla frontiera dell'immagine, che assumono in partenza un costo nullo perché ottimali sempre, per ciascun quadrilatero $Q_{i,j}$, formato dai due triangoli T_1, T_2 , i costi vengono assegnati in sincrono soltanto a quattro archi su sei, e precisamente: al lato diagonale di entrambi i triangoli (quello condiviso), al lato superiore di $Q_{i,j}$, e a quello sinistro, indipendentemente dalla loro appartenenza a T_1 o T_2 ; questo perché solo i rispettivi triangoli adiacenti sono stati già creati.

Il completamento dei costi dei lati destro e inferiore di ogni triangolo creato è asincrono, viene semplicemente rimandato al momento della creazione del triangolo adiacente. Per la precisione, nel processare $Q_{i,j}$, dopo aver calcolato i costi degli archi superiore, sinistro, e diagonale, ciascuno di essi viene registrato in entrambi i triangoli che contengono l'arco; completando così la valutazione dell'arco inferiore del quadrilatero $Q_{i-1,j}$ e dell'arco destro di $Q_{i,j-1}$. Si dimostra per induzione lungo le righe e lungo le colonne, che, in questo modo, tutti gli archi della triangolazione iniziale saranno stati valutati.



(a) il quadrilatero iniziale, sono indicati i 5 archi del quadrilatero. **(b)** singolo scambio di diagonale

3.4.2 Fase Iterativa

Ogni **Iterazione** necessaria per sostituire la triangolazione corrente con una di costo minore, fino ad ottenere una triangolazione Localmente Ottimale, si svolge così:

Scorrendo uno per uno tutti i triangoli di Nt il cui flag opt sia resettato, e per ognuno di essi scorrendo ciascun lato, sia esso l'arco e , si individua un quadrilatero formato dai due triangoli che condividono e .

Se il quadrilatero è convesso, si valuta la possibilità di sostituire e con la diagonale opposta, o quella di effettuare due contemporanei scambi di diagonale col seguente passo di **lookahead**:

- Se il singolo scambio di diagonale fa decrescere il costo della somma dei cinque archi direttamente coinvolti nel quadrilatero (vedi figura 3.3a), lo si effettua e si procede lungo Nt (figura 3.3b);
- Altrimenti, si prendono in considerazione anche i quattro triangoli adiacenti al quadrilatero (di conseguenza gli archi coinvolti saranno 13 come si vede in figura 3.4a) e:

Se lo scambio congiunto della diagonale e (con la sua diagonale opposta), e di uno dei quattro lati esterni (con la propria nuova diagonale opposta), comporta un calo della somma dei 13 costi, si fanno entrambi gli scambi e si procede lungo Nt .(figura 3.4).

Nel caso di un lato i cui due triangoli associati non formano un quadrilatero convesso proprio (vedi figura 3.5), la computazione prosegue senza prenderlo in esame.

Le figure 3.4b, 3.4c, 3.4d mostrano i tre possibili cambiamenti della figura 3.4a, il lato escluso per mancata convessità è evidenziato in figura 3.5.

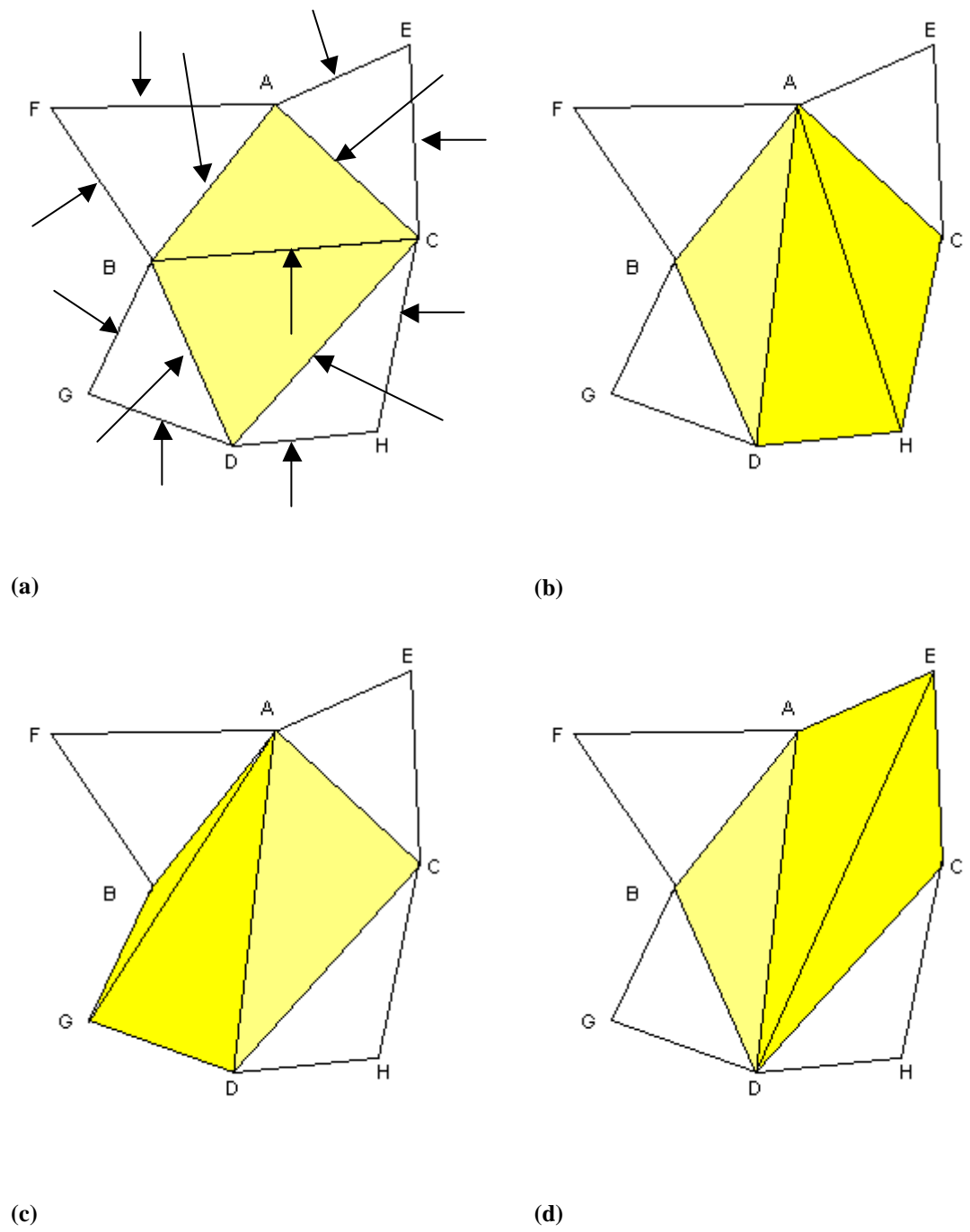


Fig. 3.4 : (a) i 13 archi del lookahead. (b), (c), (d) i casi possibili di doppio swap sulla triangolazione (a)

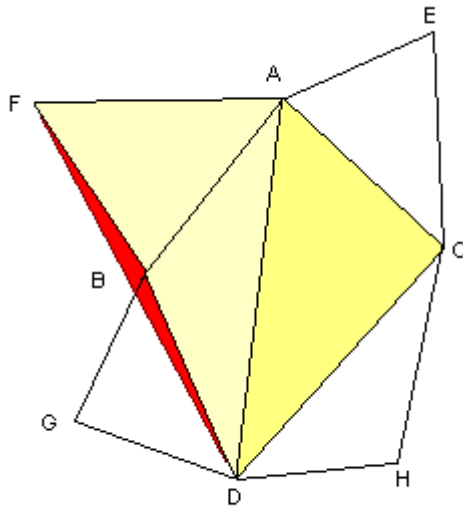


Fig. 3.5 : dopo il primo scambio di diagonale, i due triangoli ABF e ABD non formano un quadrilatero convesso

3.4.3 Ottimizzazione

Nel codice di DDZ sono stati inclusi diversi accorgimenti volti all'ottimizzazione computazionale, primo fra tutti il passo di lookahead, introdotto da X.Yu, B.S.Morse e T.W.Sederberg in [3], ampiamente descritto al paragrafo precedente.

Un altro importante contributo all'efficienza computazionale è stato ottenuto introducendo il flag *opt*:

Si consideri lo scambio di diagonale rappresentato in figura 3.3.

Al momento in cui esso viene effettuato, l'arco AD è localmente ottimale.

Sia Q il quadrilatero evidenziato. I due nuovi triangoli ABD e ACD hanno rimpiazzato i vecchi, ma Q mantiene la forma.

Si considerino, adesso, i quattro triangoli adiacenti a Q . Nell'eventualità che, fino alla fine della computazione, nessuno di essi fosse sostituito, sia il costo di AD sia del suo opposto BC resterebbero invariati, dunque ABD e ACD rimarrebbero certamente nella triangolazione localmente ottimale T_f data in output da DDZ. Pertanto, finché non avviene alcuna modifica nei triangoli adiacenti, perché sprecare tempo ad esaminare l'attuale configurazione di Q ?

La soluzione è segnalare la situazione di temporanea locale ottimalità in entrambi i triangoli creati da uno scambio, tramite settaggio del flag *opt*, e contemporaneamente resettarlo nei triangoli adiacenti a quelli coinvolti nello scambio, per mantenere l'informazione sempre aggiornata in tutta la triangolazione Nt .

L'aggiunta di questa ottimizzazione è stata testata confrontandone i risultati ed il tempo di computazione, con la versione non ottimizzata dell'algoritmo, differente da quella appena descritta soltanto nell'utilizzo del flag *opt*, cioè nel fatto che scorre, ad ogni iterazione, tutti i triangoli dell'array Nt , nessuno escluso. A dispetto del risparmio evidente di risorse temporali, l'unica differenza rilevata è l'ordine in cui i vari triangoli vengono esaminati, e questo, come si è già detto al paragrafo 2.3, non ne pregiudica la condizione di ottimalità locale.

3.5 PSNR

Lo schema 3.1, corredato di visualizzazione dell'immagine ricostruita ad ogni passo iterativo, permette una prima valutazione soggettiva del metodo, in confronto con l'interpolazione bicubica, nonché dei miglioramenti apportati da ciascuna iterazione rispetto alla precedente, e di altre caratteristiche che, però, da sole, non permettono di estrapolare statistiche consistenti e oggettive.

Per valutare gli algoritmi di zooming bisogna avere un metodo di valutazione della qualità visiva degli ingrandimenti; sarebbe necessaria una valutazione qualitativa di molte immagini ingrandite da parte di un grande numero di osservatori umani, ma, così facendo, non si potrebbe assicurare precisione e oggettività.

Ci sono due classi di metodi di valutazione oggettiva della qualità di ricostruzione: la prima tramite misure matematiche, come il Mean Square Error (MSE), il Peak Signal to Noise Ratio (PSNR), il Root Mean Square Error (RMSE), il Mean Absolute Error (MAE), e il Signal to Noise Ratio (SNR).

La seconda classe di metodi di misurazione prende in considerazione le caratteristiche del sistema visivo umano (HVS) nell'intento di includere misure di qualità percettiva.. Sfortunatamente, nessuna delle complicate metriche oggettive di questa classe proposte in letteratura si è dimostrata più vantaggiosa delle semplici misure matematiche.

La misura di qualità generalmente usata per convalidare esperimenti di image enhancement è il PSNR (massimo rapporto segnale-rumore).

Per questo motivo l'applicazione DDZ fornisce, come strumento per la valutazione di ogni esperimento, il calcolo automatico del PSNR tra l'immagine sorgente e quella ricostruita.

Bisogna, comunque, ricordare che, come tutte le altre misurazioni oggettive note, il valore PSNR tra due immagini non sempre rispecchia rigorosamente l'impressione visiva: a volte, immagini che appaiono più gradevoli di altre, o più simili all'originale, producono valori minori (cioè peggiori) di PSNR.

Sia I_1 l'immagine originale di dimensioni $M \times N$, e sia I_3 l'immagine $M \times N$ ricostruita con il procedimento descritto dallo schema 3.1.

L'errore quadratico medio (MSE) tra le due immagini I_1 e I_3 è definito come:

$$\text{MSE} = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N [I_1(x,y) - I_3(x,y)]^2 \quad (\text{Eq. 3.4})$$

Il PSNR tra le due immagini è computato scalando il MSE secondo il range dell'immagine:

$$\text{PSNR} = -10 \log_{10} \frac{\text{MSE}}{S^2} \quad (\text{Eq. 3.5})$$

Ovvero

$$\text{PSNR} = 10 \log_{10} \left(\frac{S^2}{\text{MSE}} \right)$$

Dove S è il massimo scarto possibile tra due valori di colore.

Osservando l'equazione 3.4, si vede che il MSE aumenta all'aumentare delle discrepanze tra le due immagini, infatti, ciascuna differenza tra due pixel corrispondenti, elevata al quadrato, è sempre positiva, quindi contribuisce positivamente nell'intera sommatoria, ed il fattore moltiplicativo $1/MN$ è costante e positivo anch'esso.

Inoltre, dalla formula del PSNR, (Eq 3.5), è evidente che la proporzionalità con il MSE è inversa, poiché 10 e S^2 sono costanti positive, e il logaritmo su dominio positivo è una funzione monotona crescente; il segno meno inverte la relazione.

In definitiva, maggiore risulta il valore del PSNR, e migliore è l'approssimazione numerica dell'immagine I_3 rispetto alla I_1 .

3.6 Immagini a colori

In ogni triangolo sono mantenute separatamente, sottoforma dei loro coefficienti polinomiali, le espressioni dei polinomi interpolatori lungo i tre canali R, G, e B. Se indichiamo l'equazione lineare del piano che passa per i tre valori di rosso dei vertici di un triangolo, con:

$$P(x,y) = a_r x + b_r y + c_r$$

La quantità di rosso interpolata per ciascun pixel (x,y), interno al triangolo, è

$$z_r = a_r * x + b_r * y + c_r ;$$

analogamente, per il verde e il blu

$$z_g = a_g * x + b_g * y + c_g ;$$

$$z_b = a_b * x + b_b * y + c_b ;$$

di conseguenza tale pixel assumerà il valore

$$RGB (z_r , z_g , z_b) ;$$

Per quanto riguarda la *funzione-costo*, invece, le singole componenti confluiscono nel costo unico:

$$\text{cost} = 0.21267 * R_{\text{cost}} + 0.71516 * G_{\text{cost}} + 0.07217 * B_{\text{cost}}$$

La scelta dell'arco è fatta su una combinazione lineare dei tre costi, analoga all'espressione per ricavarne la luminanza (è possibile anche lavorare direttamente sulla luminanza, cfr. [3], ma gli stessi autori dell'articolo hanno sperimentato leggeri miglioramenti nel modo qui scelto).

L'applicazione supporta sia segnali a colori sia a toni di grigio, interpretando questi ultimi come immagini a colori, i cui pixel hanno la stessa intensità originale distribuita uniformemente in tutti e tre i canali di colore.

Capitolo 4 : Sperimentazione

La sperimentazione ha avuto l'intento di testare la correttezza dell'algoritmo DDZ qui presentato, rilevarne i punti di forza e di debolezza, e di confrontarne i risultati con diversi buoni algoritmi di zooming noti, descritti al paragrafo 1.4.

Si è rilevato che l'andamento medio del valore PSNR delle iterazioni intermedie è quasi sempre crescente, con variazioni sempre minori al susseguirsi delle iterazioni, fino a convergere ad un valore costante (cfr. 4.2.1), e che DDZ produce immagini mediamente più simili all'originale rispetto all'interpolazione bicubica, con valore PSNR maggiore.

In figura 4.1 si vede chiaramente come DDZ produca una immagine più coerente con l'originale, introducendo molto meno smoothing, rispetto alla ricostruzione bicubica.



Fig. 4.1 : Confronto visivo dei risultati: DDZ introduce meno smoothing.

4.1 Categorie e dimensione del DataBase

Il campione statistico, utilizzato per la sperimentazione di DDZ, e degli altri algoritmi di zooming, è costituito da cento immagini di tipologie differenti, suddivise in quattro categorie, a seconda del tipo di soggetto rappresentato e del comportamento analizzato. Si è, infatti, riscontrato, nei campioni appartenenti a ciascuna categoria selezionata, un codominio comune, ed un andamento simile dei valori PSNR durante le iterazioni dell'esecuzione.

Le categorie stabilite, sono:

- Persone: fotografie di persone, soprattutto visi, più o meno in primo piano.
- Paesaggi: paesaggi naturali, montagne, alberi, cascate, ecc.
- Testo: Suddiviso in due sottocategorie:
 - Testo stampato. Acquisito con sensori di produzione della STMicroelectronics, presso la quale è stata svolta la sperimentazione.
 - Scrittura a mano. Autografi, lettere, manoscritti, scansiti da fonti cartacee
- Varie: immagini standard, tipicamente adoperate in materia; fotografie di oggetti, ecc.

Ciascuna categoria contiene venticinque immagini, così da permettere una valutazione equa dell'apporto di ciascuna categoria, nei risultati riguardanti l'intero campione.

4.2 Analisi dei risultati del testing.

L'analisi dei dati sperimentali, si è svolta in due direzioni:

1. Lo studio della effettiva correttezza dell'algoritmo iterativo, analizzando i miglioramenti apportati alla ricostruzione dell'immagine, determinati da ciascuna iterazione, con deduzione sperimentale del numero minimo di iterazioni necessarie alla stabilizzazione dell'esecuzione.
2. Il confronto dei risultati finali, con i risultati di altri algoritmi di zooming, testati sullo stesso campione di input.

4.2.1 DDZ – andamento del PSNR durante le iterazioni.

Dallo studio sperimentale statistico, svolto sul database di immagini descritto al paragrafo 4.1, del comportamento dell'algoritmo DDZ, man mano che l'esecuzione procede, si sono potute dedurre alcune importanti osservazioni sull'andamento della qualità di approssimazione (secondo la metrica del PSNR cfr. 3.5).

Per ciascuna immagine sono state eseguite svariate iterazioni monitorate sotto due aspetti incrociati:

- 1) Si è calcolata, ad ogni passo, una misura della discrepanza tra l'immagine risultante e quella iniziale (il PSNR, cfr. 3.5); e si è registrata la differenza tra la misura corrente e la precedente (indice di quanto l'iterazione sia stata stabilizzante). Tale differenza, nel tendere a zero, ha evidenziato un andamento crescente quasi ovunque.
- 2) Contemporaneamente, la verifica visiva dell'apporto di modifiche all'immagine, ha permesso di escludere l'eventualità di falsa-stabilizzazione, cioè coincidenze di PSNR tra due immagini consecutive ma prodotte da due triangolazioni non identiche.

Con il procedimento descritto, si è potuto verificare che il passo base (l'inizializzazione), produce una corretta ricostruzione dell'immagine, simile all'originale, e che ciascun passo iterativo successivamente eseguito, migliora visivamente l'aspetto della stessa.

Va di pari passo, non sempre ma abbastanza spesso, la misura PSNR di qualità dell'approssimazione numerica tra l'immagine ottenuta ad una certa iterazione, e l'immagine iniziale grande, precedente alla riduzione.

Dopo aver verificato che l'algoritmo tende a stabilizzarsi, e che lo fa tendendo ad aumentare l'accuratezza dell'approssimazione, è stato stimato in media, il numero massimo di iterazioni necessarie alla stabilizzazione. Se consideriamo il campione generico, costituito da tutte e quattro le categorie di immagini utilizzate, esso è, in media, pari a 4.

Ciò significa che, per ottenere la miglior ricostruzione che DDZ sia in grado di dare, o, quantomeno, una che le differisce per pochissimi pixel, basta applicare all'immagine input, un passo di inizializzazione ed al più quattro iterazioni DDZ.

A seconda della categoria del campione, queste misure presentano o meno oscillazioni, più o meno marcate, dopo le primissime iterazioni, per riprendere a crescere, poi, anche se sempre con minor pendenza, e convergere, infine, ad un valore costante (cfr. fig.4.2). In particolare, la categoria delle fotografie di persone, non presenta alcuna oscillazione, mentre nella categoria testo, la sottocategoria testo stampato, presenta le maggiori oscillazioni, tanto che le previste quattro iterazioni, calcolate in media su tutto il campione, in questo tipo di immagini, non risultano sufficienti per evidenziare il tipico andamento riscontrato in tutte le altre categorie di immagini. Sarà compito della seconda fase (cfr. paragrafo 4.3) del lavoro di ricerca che si sta svolgendo in azienda, approfondire la sperimentazione su questo genere di immagini, e concentrare la ricerca su soluzioni specifiche per tali immagini.

In figura 4.2 è mostrato il tipico andamento riscontrato durante la sperimentazione.

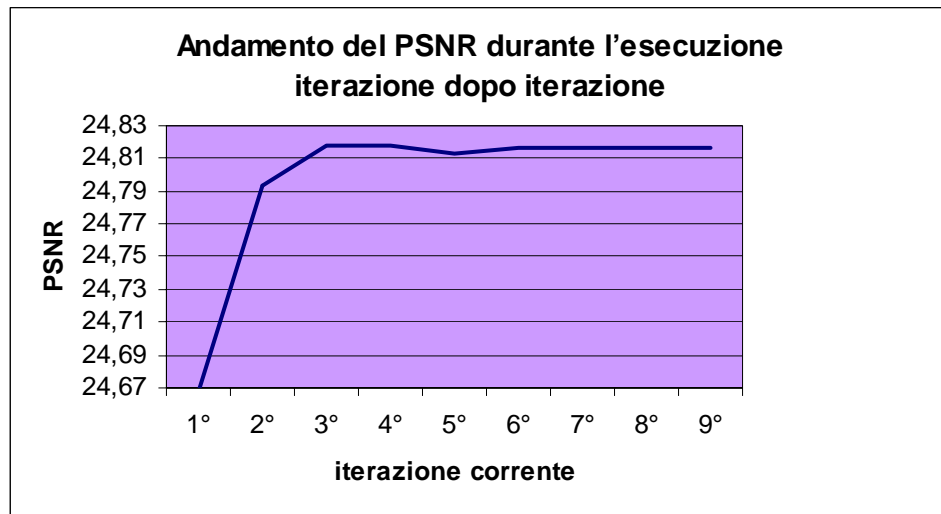


Fig. 4.2: Le iterazioni consecutive apportano miglioramenti anche nell'approssimazione numerica dei colori, l'algoritmo tende a stabilizzarsi, e già prima della quarta iterazione raggiunge un valore dal quale non si discosta più, se non di pochissimo.

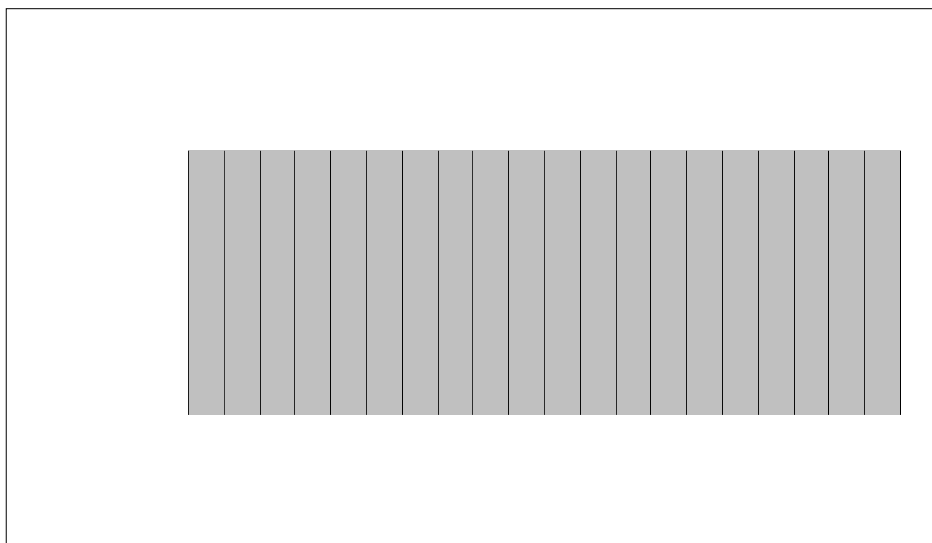
4.2.2 PSNR di vari algoritmi di Zooming

Dal testing su database di immagini di vario genere, si è rilevato che DDZ produce immagini mediamente più simili all'originale rispetto all'interpolazione bicubica, la migliore tra le tecniche classiche, e la più usata nei software commerciali. Il peggior risultato è dato, come previsto, dalla replica dei pixel. Tra gli algoritmi più recenti, analizzati, il migliore in media è risultato ReST. La figura 4.3 mostra i grafici estrapolati dal confronto dei valori PSNR prodotti da DDZ e dagli altri algoritmi elencati al paragrafo 1.4. L'algoritmo indicato col nome 'rest2' è una versione successiva di ReST.

Ciascun grafico mostra i risultati relativi ad una categoria di campioni.

Le figure 4.3.a, 4.3.b, evidenziano un comportamento migliore di DDZ rispetto alle tecniche classiche, all' algoritmo adattivo, e al BGS (LAZA), mentre ReST li supera tutti di molto. Riguardo alla figura 4.3.c, che mostra il peggior PSNR in corrispondenza dell' algoritmo DDZ, bisogna ricordare che, invece, dal punto di vista dell' effetto visivo, si è riscontrata, quasi sempre, una tendenza diversa: I paesaggi ricostruiti da DDZ appaiono meglio definiti e di aspetto più gradevole, più simile alla fotografia originale, rispetto alle tecniche classiche di interpolazione.

Questa discrepanza di risultati non stupisce, poiché, come spiegato nell' introduzione, DDZ è un algoritmo che tende a minimizzare gli artefatti e gli effetti visivi sgradevoli dello zooming, più che l' approssimazione numerica dei pixel mancanti.



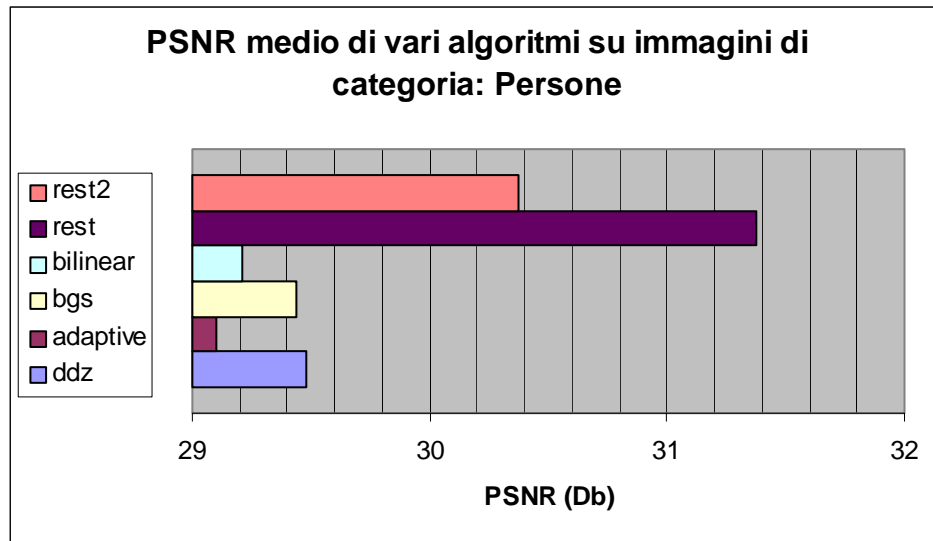


Fig. 4.3.b : Su fotografie di persone, dai primi piani alla figura intera, DDZ dà risultati migliori delle tecniche classiche, e degli algoritmi bgs e adaptive. Rest produce i PSNR migliori.

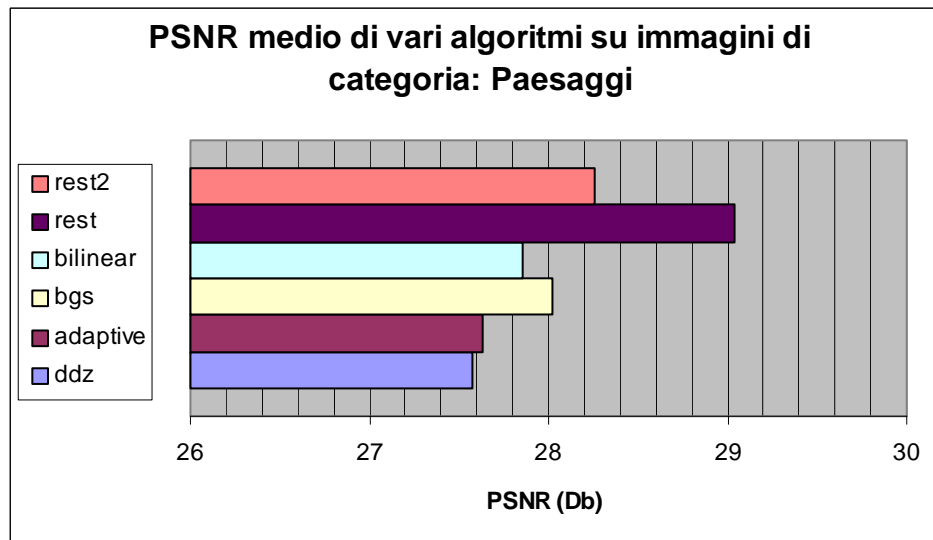


Fig. 4.3.c : Su fotografie panoramiche, DDZ produce i valori PSNR peggiori. Rest produce i PSNR migliori.

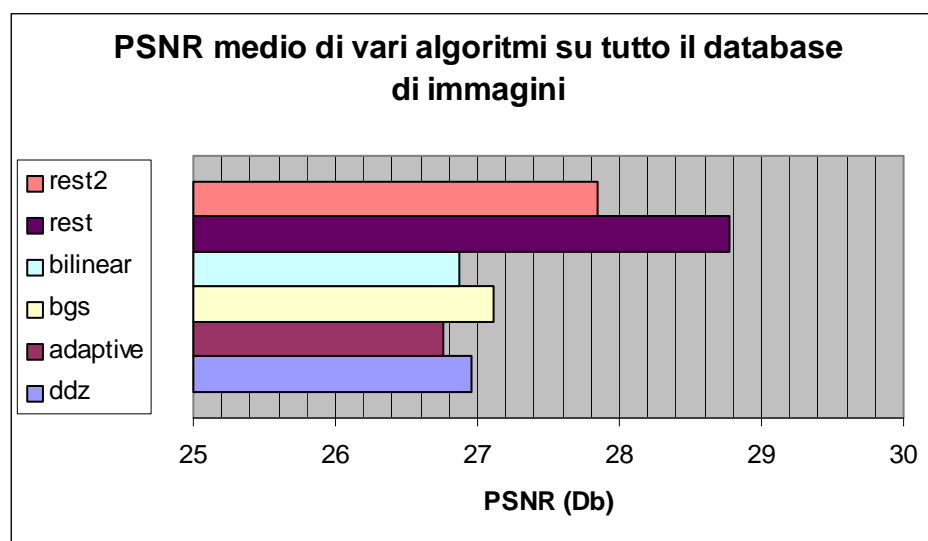


Fig. 4.3.d: Media dei risultati di tutte le categorie di immagini.

Per quanto riguarda l'algoritmo Pixel Level Data Dependent Triangulation, di Su e Willis, la versione basic dell'algoritmo ha mostrato caratteristiche paragonabili all'inizializzazione di DDZ, sia per la somiglianza visibile del relativo output su medesimi input, sia per la ridotta entità dello scarto fra il PSNR delle due serie di esperimenti, con differenze dell'ordine di 10^{-3} , tra esse. Un esempio di paragone fra i due comportamenti simili è mostrato in figura 4.4, il grafico (a) confronta i due algoritmi su tredici immagini miste tratte dal database costruito, il grafico (b) ne evidenzia i relativi scarti, per poterli notare si è dovuta ingrandire di molto la scala.

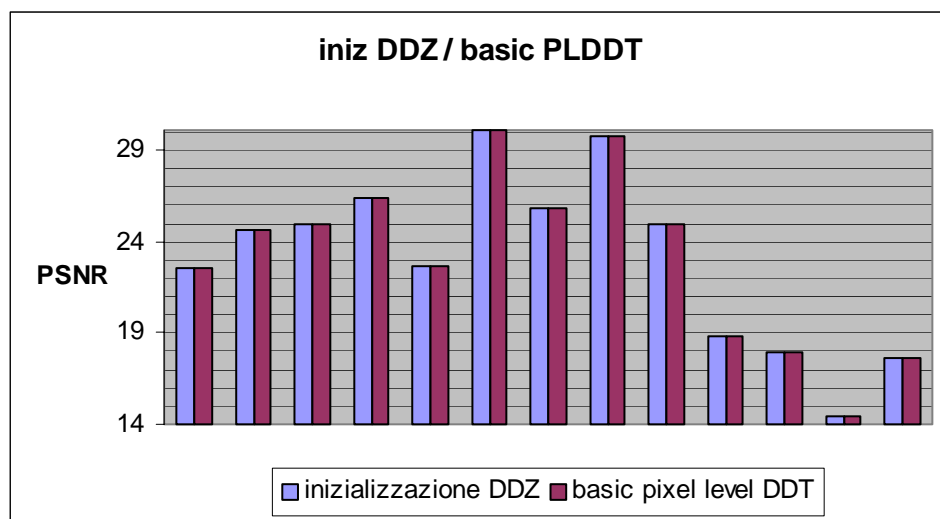


Fig. 4.4.a: La versione basic della Pixel level DDT produce risultati molto simili a quelli dati dalla sola inizializzazione DDZ.

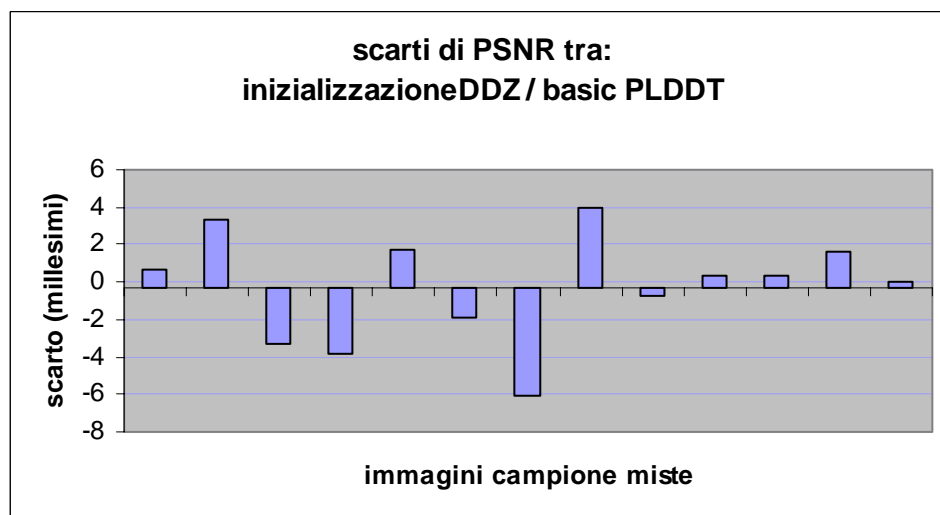


Fig. 4.4.b: Non c'è una netta prevalenza di uno dei due metodi sull'altro: gli scarti sono minimi e sono distribuiti in entrambe le direzioni.

Bibliografia

- [1] F.Preparata, M.I.Shamos, "Computational Geometry: An Introduction", Spring-Verlag, 1985.
- [2] N.Dyn, D.Levin, S.Rippa, "Data Dependent Triangulations for Piecewise Linear Interpolation" IMA Journal of Numerical Analysis, Vol. 10, pp. 137-154, 1990.
- [3] X.Yu, B.Morse, T.W.Sederberg, "Image Reconstruction Using Data-Dependent Triangulation", IEEE Transactions on Computer Graphics and Applications, pp. 62-68, Maggio/Giugno 2001.
- [4] S.Battiato, G.Gallo, F.Stanco, "Image enhancement techniques", tesi di Dottorato, Università degli studi di Catania.
- [5] D.Su, P.Willis, "Image Interpolation by Pixel Level Data-Dependent Triangulation", sottoposto al Computer Graphics Forum, 10/2002.
- [6] S.Dube, Li Hong, "An Adaptive Algorithm for Image Resolution Enhancement", AST San Diego Lab, STMicroelectronics, Inc.
- [7] S.Battiato, G.Gallo, F.Stanco, "A locally adaptive zooming algorithm for digital images", Image and Vision Computing Vol. 20, 2002, pubblicato da Elsevier Science B.V.
- [8] S.Battiato, G.Gallo, M.Mancuso, G.Messina, F.Stanco, "Analysis and Characterization of Super-Resolution Reconstruction Methods", STMicroelectronics, Catania (Italy).
- [9] S.Lertrattanapanich, N.K.Bose, "High Resolution Image Formation From Low Resolution Frames using Delaunay Triangulation", IEEE Transactions on Image Processing, Vol. 11, No. 12, 12/2002.