POLITECNICO DI MILANO

Content Aware Image Resize

Prof. Pierluigi Taddei

Autori:
Alessio C. Bonsignore - 711306
Stefano Anelli - 711309

Anno Accademico 2007-2008

Documentazione relativa al progetto realizzato durante il corso Image Synthesis and Analysis tenuto dal prof. Vincenzo Caglioti

SOMMARIO

INTRODUZIONE

CONTESTO

L'evoluzione nelle telecomunicazioni ormai consente di rimanere connessi ad internet sempre e ovunque. Mentre un tempo si utilizzava solo lo schermo del proprio PC Desktop per visualizzare i contenuti del web, ora è possibile connettersi con un'innumerevole quantità di dispositivi, compresi portatili, UMPC, palmari e cellulari. Questo significa dover adattare le stesse risorse internet su schermi di dimensioni estremamente diverse. Cellulari e palmari partono da una risoluzione di 320x240. Uno schermo da 30", che non è più una soluzione così estrema come si potrebbe pensare, adotta una matrice di pixel da 2560x1600.

DESCRIZIONE DEL PROBLEMA

Adattare gli stessi contenuti multimediali su schermi di diverse dimensioni spinge l'attenzione in svariati sottoproblemi. In questo progetto ci si concentrerà sulla gestione di foto ed immagini.

Solitamente l'autore dei contenuti dispone di immagini e foto ad alta risoluzione. L'idea generale è quella di effettuare un ridimensionamento della matrice di pixel che compone l'immagine. Normalmente però quest'operazione si limita a comprimere tutti i pixel uniformemente in modo che si adattino alla nuova dimensione. Questo è già un inizio, ma porta con sé spiacevoli inconvenienti quali la deformazione dell'immagine se il rapporto larghezza/altezza non si conserva (vedi Fig.1) e la perdita di molti dettagli. Ma cosa si intende per *dettagli*? Questa è una domanda che si sono posti i ricercatori di *Analisi e Sintesi di Immagini* e si cercherà di darne una risposta nel corso di questo articolo.

FIGURA 1 L'immagine a sinistra è una fotografia d'esempio. Si è scelto il formato 16:9 solito delle trasmissioni televisive. L'immagine al centro è stata ridimensionata in larghezza con un semplice algoritmo in modo che si adattasse al formato 4:3 più tradizionale. E' evidente come il volto risulti deformato. Un ridimensionamento più piacevole alla vista è mostrato a destra. Un algoritmo che generi un risultato di questo tipo è argomento di studio di questo progetto.

La preoccupazione di dover trovare una soluzione alternativa più furba al problema spinge verso la necessità di realizzare una tecnica di ridimensionamento che consideri anche il contenuto stesso dell'immagine. Questa è la chiave del problema. Non si possono considerare tutti i pixel allo stesso modo. È indispensabile trovare una politica che assegni ai pixel un *grado di importanza*. Solo così l'algoritmo sarà in grado di costruire una nuova immagine mantenendo il più possibile i pixel importanti a discapito di quelli ritenuti privi di *informazione*. Per capire meglio cosa si intende, si consideri ancora una volta la Figura 1. È chiaro come lo sfondo violetto sia trascurabile in questa fotografia. I pixel da considerare importanti sono quelli che compongono il volto della ragazza. Un algoritmo accorto dovrebbe capire questa differenza e ridimensionare l'immagine tagliando lo sfondo nella parte sinistra e mantenendo il più possibile i pixel sulla destra dove si trova l'informazione.

SCOPO DEL PROGETTO

L'ipotesi di partenza è quella di possedere delle immagini ad alta risoluzione adatte per gli schermi più grandi. Data questa premessa ci poniamo come obiettivo la realizzazione di un algoritmo di ridimensionamento di tipo content-aware che data un immagine, dopo averne analizzato il contenuto, sintetizzi una seconda immagine di dimensioni

inferiori che mantenga il più possibile le informazioni dell'immagine di partenza sacrificando aree di minore importanza. L'immagine ottenuta può così essere utilizzata anche sugli schermi più piccoli. Questa è la soluzione che è stata pensata per il problema iniziale.

Sarà necessaria una prima fase di analisi del problema seguita dalla progettazione di una tecnica di content-aware. Verrà infine realizzata una funzione Matlab che segue quest'algoritmo.

PROGETTO

ANALISI DEL PROBLEMA

ARTICOLI

Il progetto è iniziato con la lettura di alcuni articoli consigliati dal prof. Taddei. Questi sono:

- **Seam Carving for Content-Aware Image Resizing** di Shai Avidan Ariel Shamir
- **Non-homogeneous Content-driven Video-retargeting** di Lior Wolf, Moshe Guttmann, Daniel Cohen-Or

Come si vedrà in seguito, l'algoritmo si basa fortemente sulla tecnica studiata nel secondo articolo che viene da noi analizzata e modificata.

FUNZIONE ENERGIA

SIGNIFICATO

Com'è stato spiegato nel capitolo introduttivo, il punto di partenza di queste nuove tecniche di ridimensionamento content-aware è associare un peso ad ogni pixel dell'immagine. Questo peso indica l'importanza del pixel nell'immagine basata sull'informazione visiva che consegna all'osservatore. Più il pixel è rilevante, più questo si trova in una zona ricca di dettagli preziosi che bisogna cercare di mantenere il più possibile per non alterarne il contenuto e il significato dell'immagine. Solo valutando questi pesi sarà possibile, in una fase successiva, effettuare il ridimensionamento vero e proprio. In seguito ci si riferirà ai pesi come *energia* o *saliency*. Questi valori vengono organizzati in una matrice di interi positivi della stesse dimensioni dell'immagine di partenza. Più il valore sarà alto, più il pixel associato avrà un energia alta. Non importa che la matrice sia normalizzata e che contenga quindi valori compresi tra 0 e 1. Può comunque essere scalata in qualsiasi momento per migliorare la precisione di calcolo.

Chiarito il significato di energia, la *funzione energia* riceve in ingresso un immagine in scala di grigi e restituisce una matrice di pari dimensioni in cui ogni elemento associa un valore di energia ad ogni pixel dell'immagine in base ad una politica prestabilita.

Inoltrandosi nei dettagli del progetto necessariamente iniziano a presentarsi le prime limitazioni. La funzione energia da noi implementata accetta in ingresso un immagine in scala di grigi. Chiaramente è possibile utilizzare immagini a colori, a patto di possedere un algoritmo che preventivamente la converta in scala di grigi. Questo è un'ovvia limitazione, ed in seguito cercheremo i dettagli basandoci solo sulla luminosità dei pixel e non sulla tonalità o la saturazione. Ad esempio, se avessimo l'immagine composta da aree con la stessa luminosità ma colore molto diverso, non riusciremo a trovarne i contorni dopo la conversione. Il progetto considera questa limitazione accettabile dato che empiricamente è molto difficile avere fotografie reali il cui contenuto ha una luminosità molto uniforme.

Nei paragrafi successivi verranno analizzate più politiche per la realizzazione della funzione energia. E' però già possibile fare una considerazione importante. Ogni singola politica valuta i pixel in base ad un criterio diverso, cercando di trovare le zone dell'immagine con più informazione. Tuttavia, ogni singola politica è importante e non bisogna considerarle contrastanti. Bisogna cercare di utilizzarle insieme in modo da ottenere i benefici di ognuna. Nessuna politica usata da sola riesce a garantire una copertura per tutti i dettagli. La soluzione migliore è usarle insieme. Più in particolare, una volta ottenute le matrici con l'energia, basterà effettuare una somma pesata delle stesse.

Gradient Magnitude

La prima fonte di informazioni è chiaramente l'intensità del gradiente che si ricava direttamente dalle derivate parziali eseguite sull'immagine in scala di grigi. Per ottenerla basta eseguire una convoluzione tra la matrice dell'immagine ed un operatore. Esistono diverse matrici operatore di cui Sobel è uno dei più famosi. Tuttavia, per il nostro progetto abbiamo utilizzato il più classico operatore di Prewitt:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad S_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * I \qquad S = \sqrt{S_x^2 + S_y^2}$$

Nelle equazioni I rappresenta la matrice dell'immagine in scala di grigi, mentre S_x e S_y le due derivate parziali. Per ottenere l'intensità è sufficiente sommare la radice quadrata dei quadrati delle due derivate parziali, facendo attenzione ad eseguire un elevamento a potenza elemento per elemento e non in forma matriciale.

Solitamente questo basterebbe per ottenere l'approssimazione del gradiente. Nella nostra specifica applicazione però sorge un problema. Gli elementi sul bordo della matrice sono stati ottenuti con valori di padding che quindi non hanno nessuna relazione con l'immagine. Come vedremo questo crea un serio problema al momento di sintetizzare la nuova immagine, e per questo è necessario modificare questi valori. Ancora una volta si possono utilizzare più strategie. E' possibile tagliare i bordi ottenendo però una matrice più piccola rispetto all'immagine di partenza; oppure è possibile eseguire altre convoluzioni con operatori diversi che non utilizzano il padding; la nostra soluzione sostituisce i valori sul bordo con la media di tutti gli elementi della matrice. Inizialmente si era pensato di utilizzare 0 come valore da sostituire, ma stranamente questo valore portava a dei risultati inaspettati durante la sintesi.

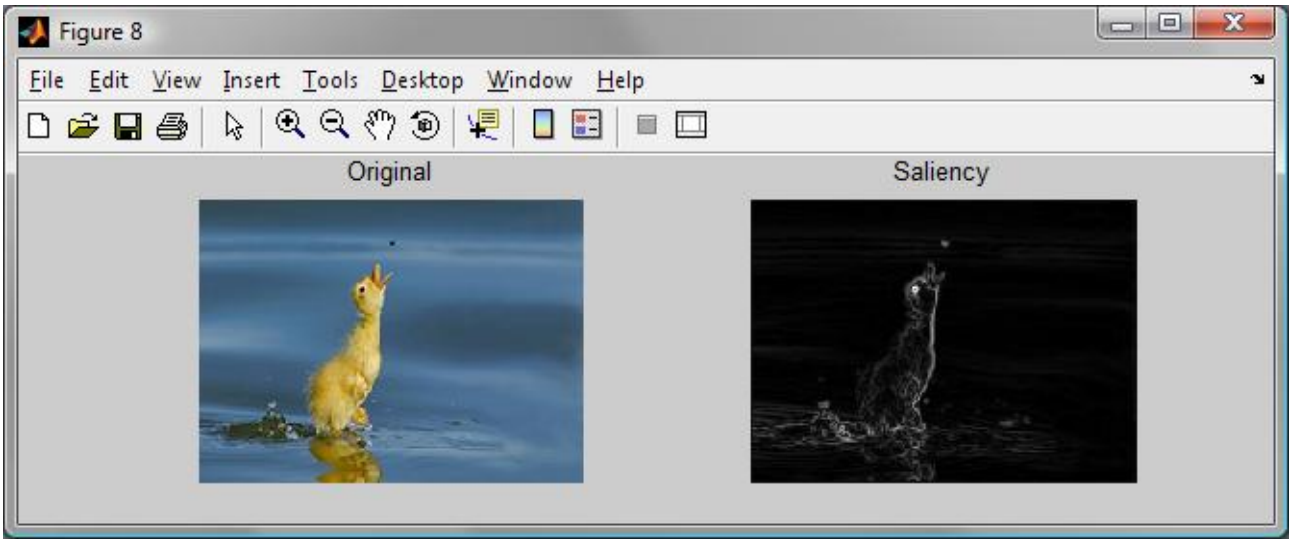

FIGURA 2 L'immagine a destra mostra in scala di grigi come appare la funzione energia ottenuta dall'intensità del gradiente dell'immagine a sinistra. Si può notare come un repentino cambiamento di colore determini un alto valore nella matrice Saliency e di conseguenza un colore chiaro nell'immagine. Al contrario aree sfumate come lo sfondo portano a valori vicini a zero ed un conseguente colore nero.

Face Detection

E' molto facile trovare fotografie in cui appaiono dei volti. Essendo inoltre il volto umano molto conosciuto, qualsiasi difetto o deformazione risalterebbe all'occhio dell'osservatore. Per questo motivo è consigliabile utilizzare un qualunque algoritmo di face detection per individuare la posizione e la dimensione dei volti nell'immagine per poter così preservare quelle zone il più possibile. A questo proposito è possibile marcare una zona disegnando nella matrice una circonferenza con centro nel centro del volto e che si dissolve in modo sfumato allontanandosi da esso. Il diametro della circonferenza deve dipendere dall'estensione del volto.

Nel secondo articolo citato si fa riferimento all'algoritmo di face detection di Viola e Jones che si basa sull'addestramento di una rete neurale. L'algoritmo è presente online e può essere utilizzato liberamente. Nel

progetto sono stati eseguiti con buoni risultati dei test sul face detection utilizzando proprio questo algoritmo, ma non è stato mantenuto nella versione finale perché si è preferito utilizzare un approccio differente e più generale che verrà discusso in seguito.

FEATURES DETECTION

Questa politica nasce da un idea del professor Taddei e cerca di andare incontro alle stesse necessità che portano all'uso di un face detection senza però rimanere limitati dal particolare contesto dei volti umani. Sostanzialmente l'idea è quella di localizzare nell'immagini le features, i corners, i punti di rilievo e di considerare le aree tra questi punti molto rigide, in modo che non vengano deformate durante il ridimensionamento. Esistono molti modi per trovare le coordinare delle features in un immagine. Nel progetto viene adottato l'Harris Corner Detection, che oltre alla posizione consente di avere una stima dell'importanza delle features (leggere il codice sorgente per ulteriori chiarimenti). In alternativa è possibile appoggiarsi a metodi anche più raffinati, come ad esempio SIFT (Scale-Invariant Features Transform). Questo algoritmo è stato provato nel progetto ma in ultima analisi ci si è accorti che il più semplice Corner Detection è già sufficiente.

La ricerca delle features ha una sua validità se si pensa come queste siano tra i punti che danno più informazione in un immagine. Mantenerle inalterate, e mantenere le proporzioni tra loro invariate, comporterebbe un sicuro vantaggio nella resa del risultato finale. Bisogna però fare molta attenzione a trovare una correlazione tra le features. E' sbagliato pensare che siano tutte correlate tra loro. Solo le features di uno stesso soggetto sono in relazione e devono quindi essere mantenute inalterate nelle proporzioni. Trovare i soggetti nelle immagini è un compito assai arduo e ad oggi non si è ancora riusciti a trovare una tecnica che sia soddisfacente in tutte le situazioni, anche perché è un problema che valica il limite della semantica. Gli algoritmi che tentano di eseguire questo compito sono chiamati di

FIGURA 3 Qui vengono riportate delle immagini con delle possibili segmentazioni associate. Come si può intuire il problema è molto complesso, e anche solo scegliere il grado di segmentazione che si vuole ottenere risulta problematico. Notare ad esempio come nella prima immagine si possa decidere di unire o meno le singole assi della staccionata; oppure come nella seconda immagine si possa considerare lo sfondo unico, diviso in due, oppure molto frastagliato.

segmentazione. Come spesso accade, si sono ottenuti buoni risultati considerando più fattori contemporaneamente e cercando di utilizzare la potenza delle reti neurali.

Purtroppo in rete non si trovano funzioni disponibili che siano così avanzati. Per il progetto sono stati provati alcuni algoritmi che però sono hanno portato a buoni risultati. Per questo motivo si è cercato di aggirare il problema con soluzioni alternative.

Una volta individuate le features nell'immagine, per suggerire all'algoritmo la loro posizione si possono utilizzare due approcci differenti. Il primo è quello di alterare direttamente i valori di energia nella matrice S incrementando gli elementi delle zone interessate. In particolare è possibile tracciare dei segmenti tra le features in modo da formare la griglia con triangoli di Delaunay. Per tracciare i segmenti è noto il semplice algoritmo di Bresenham, che se necessario può essere seguito da una convoluzione con un filtro dilation per rendere più spessi i segmenti. Questi segmenti suggeriscono all'algoritmo che quelle zone sono importanti e non verranno modificate. Questo concetto risulterà più chiaro nel prossimo capitolo. Prima di tracciare i segmenti che uniscono le features, bisogna però ricordarsi della correlazione tra i soggetti dell'immagine. La soluzione migliore sarebbe quella di eseguire sull'immagine una funzione di segmentation e unire tra loro solo features che si trovano nello stesso gruppo. Purtroppo, come è stato accennato, è molto difficile realizzare una funzione di segmentation precisa, quindi per il progetto si è cercato di aggirare il problema. Si è scelto di disegnare i segmenti con intensità diverse nell'ipotesi che features lontane siano poco correlate rispetto a features vicine tra loro. Avremo quindi valori tendenti allo 0 per features lontane come la metà della massima dimensione dell'immagine, e valori crescenti man mano che la distanza di assottiglia. Un esempio di questa politica è mostrato in figura.

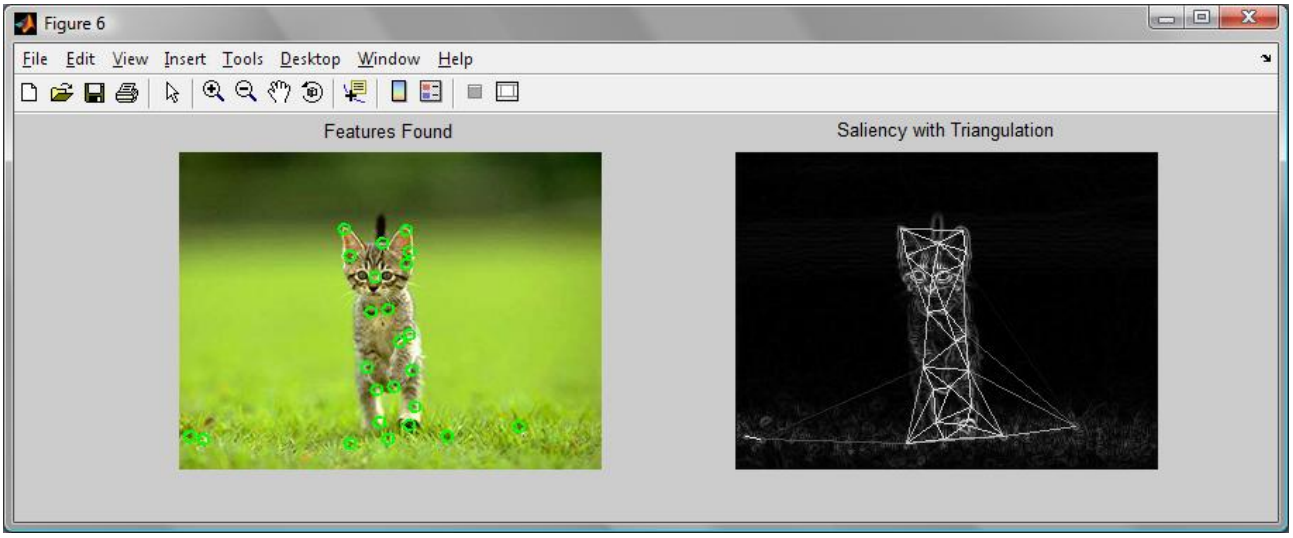

FIGURA 4 Quest'esempio mostra il risultato ottenuto dalla ricerca delle features e come queste siano utilizzate per modificare la funzione energia. Nell'immagine a sinistra è stato utilizzato l'Harris Corner Detection individuando le features che sono state evidenziate da dei cerchietti verdi. Nell'immagine a destra viene mostrata la funzione energia. Il nero determina il valore 0 mentre il bianco il massimo valore presente. Utilizzando le features è stata disegnata la griglia di Delaunay facendo attenzione ad usare intensità diverse per segmenti di lunghezza diversa. Più questi sono lunghi, più si confonderanno col nero dello sfondo, indicando una minore importanza.

Un altro modo per evidenziare le aree interessate dalle features è quello di colorare completamente tutti i pixel in quelle zone, eventualmente sfumando in modo simile a quanto proposto nel paragrafo sul face detection. Questa tecnica non è stata esplorata nel progetto.

Queste appena descritte vanno quindi a modificare la funzione energia e rappresentano un primo approccio per suggerire all'algoritmo dove sono situate le zone interessanti. Un secondo approccio invece può essere utilizzato direttamente durante la sintesi, e per questo rimandiamo la discussione al prossimo capitolo.

SINTESI DELL'IMMAGINE RIDIMENSIONATA

In questo capitolo affronteremo il vero cuore del progetto, ovvero come sintetizzare una nuova immagine a partire da un immagine sorgente più grande. Nel capitolo precedente è stato spiegato come fosse importante una matrice che descriva l'energia dell'immagine pixel per pixel, e quali tecniche si possono utilizzare per ottenerla. Questa matrice quindi è il punto di partenza.

I due articoli citati in precedenza adottano due tecniche molto diverse, ma entrambe utilizzano la funzione energia. Il primo articolo sostanzialmente cerca di ridurre una dimensione dell'immagine eliminando ripetutamente delle linee di pixel che tagliano la stessa da un lato all'altro. Per ottenere il risultato sperato, la linea deve essere quella con energia minore in tutta l'immagine. In questo modo il taglio interesserà zone con poca informazione e che possono quindi essere sacrificate. Questa tecnica comunque non è stata utilizzata nel progetto, quindi per ulteriori dettagli si suggerisce una lettura dell'articolo stesso.

Il secondo articolo invece suggerisce una tecnica che è stata da noi implementata. L'idea è quella di costruire un sistema lineare in cui le variabili rappresentano le posizioni dei pixel. Una volta impostati tutti i vincoli necessari, la soluzione del sistema esprimerà sostanzialmente la nuova immagine sintetizzata, che potrà quindi essere costruita. Per semplicità si è scelto di lavorare solo una dimensione alla volta. Questo significa che per ridurre sia in altezza che in larghezza sarà necessario eseguire il processo due volte, uno di seguito all'altro. Questa scelta è stata adottata anche nell'articolo di riferimento. Eseguire una riduzione di entrambe le dimensioni necessita di una tecnica molto diversa.

Nei prossimi paragrafi quindi ci si concentrerà sulla riduzione della sola larghezza dell'immagine.

STRUTTURA DEL SISTEMA LINEARE

Per costruire il sistema è necessario capire come rappresentare i pixel e i loro vincoli. Viene utilizzata una variabile per ogni pixel dell'immagine. Questo comporta un gran numero di variabili e, come vedremo, un lungo tempo di elaborazione. Per un'immagine 320x240 viene costruito un sistema da 76.800 variabili. Ogni variabile determina la posizione orizzontale di un pixel. Questo significa che i pixel sul lato sinistro hanno valore 0; i pixel sul lato destro assumono un valore pari alla dimensione (alla larghezza) dell'immagine. Essendo la soluzione del sistema lineare un vettore, si è reso necessario serializzare l'immagine, per cui, data un immagine WxH, i primi W elementi del vettore soluzione rappresentano le posizioni dei pixel della prima riga. Seguiranno altri (H-1) blocchi di dimensione W. L'immagine seguente cerca di chiarire questo concetto:

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

E' mostrata un immagine 3x2, i cui pixel in entrambe le righe hanno rispettivamente posizione orizzontale 1, 2, 3. Un ipotetico vettore soluzione assumerebbe la forma indicata dalla freccia, sarà pertanto necessario ri-strutturarlo prima di sintetizzare l'immagine finale.

Come si può intuire, i pixel inizialmente hanno delle posizioni orizzontali discrete. Il primo pixel è in posizione 1, il secondo in posizione 2, ecc… . Nell'immagine che si vuole realizzare invece non c'è abbastanza spazio per mantenere queste distanze. Si decide quindi di passare al continuo, in modo da poter comprimere i pixel avvicinandoli tra loro. Ipoteticamente quindi, potremo avere i pixel in posizione 1,2 , oppure in posizione 37,41. Sarà solo in una fase successiva che verrà ricostruita l'immagine discreta facendo una media dei colori dei pixel che si trovano a condividere lo stesso posto.

$$[1 \quad 2 \quad 3] \rightarrow [1 \quad 1{,}5 \quad 2]$$

Nell'esempio, partendo da una riga larga 3 pixel, si è voluto ottenere una riga larga 2 pixel. Assumendo la stessa importanza nei pixel, abbiamo avuto una ridistribuzione uniforme dei valori che ha portato il secondo elemento in posizione 1,5 e l'ultimo elemento in posizione 2. E' facile ora intuire il significato della funzione energia. Al momento di trovare la posizione orizzontale dei nuovi pixel, in base ai vincoli specificati, si cercherà di mantenere a distanza 1 i pixel più importanti, in modo da preservare l'aspetto originale dell'immagine. Al contrario, i pixel con poca importanza si troveranno compressi in un piccolo intervallo.

Capita l'interpretazione delle variabili nel sistema di equazioni, è necessario affrontare il problema di come costruire i vincoli. Inoltre va ricordato che ogni vincolo può essere più o meno rafforzato rispetto agli altri moltiplicando tutti i coefficienti per una costante.

VINCOLI DI DISTANZA ORIZZONTALE

Il primo vincolo che viene analizzato è quello di distanza orizzontale. Come accennato precedentemente, le variabili del sistema determinano la posizione orizzontale dei pixel. Nell'immagine sorgente i pixel adiacenti si trovano tutti a distanza 1 l'uno dall'altro. Nell'immagine che si vuole ottenere invece non è più possibile mantenere questa distanza, e necessariamente alcuni pixel dovranno avvicinarsi tra loro. Tuttavia, uno degli obiettivi è proprio quello di mantenere i pixel il più possibili fedeli all'originale. I vincoli di questo tipo quindi impongono una distanza pari a 1 a tutti i pixel adiacenti, ma non basta. In questo modo infatti avremmo una mappatura dei valori omogenea in tutta la riga. Per questo motivo viene utilizzata la funzione energia che assegna pesi diversi alle varie coppie di pixel. La formula viene presentata di seguito:

$$S_{i,j}\left(x_{i,j} - x_{i-1,j}\right) = S_{i,j}$$

Per un immagine WxH avremo quindi $(\mathrm{W} - 1) \cdot \mathrm{H}$ vincoli di questo tipo.

Ad esempio, per un immagine 320x240 verranno generati 76.560 vincoli di distanza orizzontale. Ancora una volta in questo progetto ci si trova ad affrontare il problema di gestire numeri molto alti che minacciano le prestazioni. Per affrontare questo problema si è studiata la conformazione che assume la porzione di matrice dei coefficienti creata da questi vincoli e si è pensato di generarla nella sua interezza senza doverla costruire vincolo per vincolo.

VINCOLI DI ALLINEAMENTO VERTICALE

Essendo l'immagine suddivisa per righe, i vincoli verticali non sono una semplice riproposizione dei vincoli orizzontale, ma assumono un significato molto diverso. Sono chiamati infatti vincoli di allineamento e non di distanza, perché in questo caso per ogni coppia di pixel adiacenti verticalmente non viene imposta una distanza pari ad 1, ma viene richiesto di rimanere il più possibili allineati tra loro. Questo si ottiene forzando i valori posizione dei pixel nella stessa colonna ad assume un valore identico. Due pixel di due righe diverse sono infatti allineati se hanno la stessa posizione orizzontale. Il vincolo assume quindi la forma:

$$x_{i,j} - x_{i,j+1} = 0$$

Anche in questo caso sono necessari un gran numero di vincoli, più precisamente in un immagine WxH ne occorrono $(\mathrm{H} - 1) \cdot \mathrm{W}$. Di conseguenza, da un immagine 320x240 nasceranno 76.480 vincoli di allineamento verticale. Il numero è dello stesso ordine di grandezza di quello relativo ai vincoli orizzontali. Anche in questo caso si è cercato di costruire la matrice dei coefficienti da grosse porzioni, evitando l'analisi dei singoli vincoli.

Nonostante il numero di vincoli sia già elevato, occorrono ulteriori vincoli di tipo verticale. Al fine di evitare una deformazione eccessiva dell'immagine è necessario porre attenzione alla prima e all'ultima riga. Se prima ci si preoccupava dell'allineamento tra i pixel di righe vicine, con questi altri vincoli ci si preoccupa che una colonna verticale non venga piegata eccessivamente. Si pensi ad esempio ad una differenza di allineamento monotona (sempre positiva, o sempre negativa) tra i pixel di una colonna. Rispetto all'immagine sorgente, quella sintetizzata risulterebbe piegata vistosamente da un lato, come quando un rettangolo viene deformato in un parallelogramma.

Per questa ragione vengono inseriti nel sistema vincoli del tipo:

$$x_{i,1} - x_{i,\mathrm{H}} = 0$$

Fortunatamente il numero di questi vincoli è nettamente inferiore agli altri. Per un immagine WxH bastano W vincoli di questo tipo. Nel solito esempio, vengono generati 320 vincoli da un immagine 320x240.

Per i vincoli di allineamento verticale è stata utilizzata una costante moltiplicativa per tutti i coefficienti chiamata rigidità (rigidity). Utilizzando un valore pari a 1 si notano tagli troppo netti, che portano le colonne di pixel ad essere mantenuta completamente oppure ad essere cancellate. Limitando la rigidity ad esempio a 0,2 si raggiunge una flessibilità maggiore consentendo ai pixel di adattarsi meglio al contenuto dell'immagine durante la compressione.

Vincoli di nuova dimensione

Una volta definiti i vincoli generali della struttura dell'immagine da sintetizzare, bisogna definire dei vincoli estremamente importanti, che danno senso a tutto il progetto. I vincoli di nuova dimensione infatti forzano le nuove posizioni a rimanere confinate all'interno del nuovo spazio assegnato. Se prima l'immagine era larga 320 pixel, e ora si desidera ridurla a 300, questi vincoli impongono questa scelta. Per farlo basta assegnare al primo pixel di ogni riga il valore 1, ed all'ultimo pixel di ogni riga il valore della nuova dimensione scelta. Quindi si avranno:

$$\forall j \; x_{1,j} = 1 \qquad\qquad \forall j \; x_{\mathrm{W},j} = new\mathrm{H}$$

I vincoli di questo tipo sono relativamente pochi. In un immagine WxH verranno definiti $2 \cdot \mathrm{H}$ vincoli di nuova dimensione, ovvero 480 vincoli da un immagine 320x240.

Vincoli tra features

Questi vincoli sono alternativi alla tecnica descritta nel capitolo relativo alla funzione energia. Una volta individuate le features nell'immagine, la tecnica descritta precedentemente suggeriva di modificare direttamente i valori della matrice energia. In quel modo sarebbero sufficienti gli altri tipi di vincoli già descritti per considerare anche il peso delle features. Una strada alternativa prevede la possibilità di non alterare la matrice energia e passare le coordinate delle features al metodo che costruisce il sistema. Vengono quindi inseriti ulteriori vincoli che tengono conto di queste nuove informazioni.

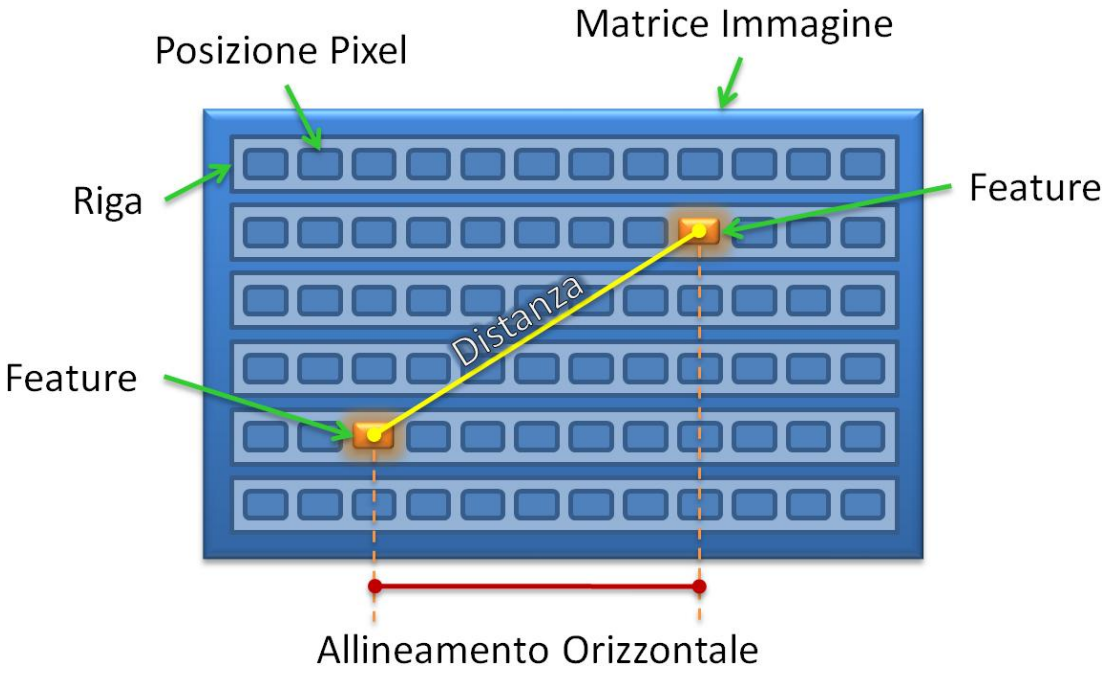

FIGURA 5 Lo schema mostra la struttura dell'immagine. Come è evidenziato, ogni riga è considerata indipendentemente dalle altre. I vincoli definiti rispetto alle features impongono l'allineamento orizzontale tra i pixel interessati, mentre il peso del vincolo è inversamente proporzionale alla loro distanza.

Per ogni segmento della triangolazione di Delaunay viene definito un vincolo che impone la proiezione della distanza tra i pixel interessati (Allineamento Orizzontale). Il peso di questi vincoli segue l'inverso della distanza effettiva tra le features, così come avveniva con l'altro metodo. Essendo la soluzione definita per righe, il vincolo terrà in considerazione solo la componente orizzontale senza intervenire su quella verticale. In seguito viene presentato uno schema che cerca di chiarire gli elementi considerati:

DIMENSIONI DEL SISTEMA LINEARE

A questo punto si è in possesso della matrice dei coefficienti del sistema lineare ottenuta impilando le sottomatrici definite da ogni tipo diverso di vincolo. Quello che si è ottenuto è un enorme sistema lineare sparso. Data un immagine in input di dimensioni WxH si ottiene un sistema con un numero di vincoli pari a:

$$\#vincoli =$$

$$= (\mathrm{W} - 1) \cdot H + (H - 1) \cdot \mathrm{W} + \mathrm{W} + 2 \cdot H =$$

$$= (2 \cdot \mathrm{W} + 1) \cdot \mathrm{H}$$

A questi si aggiungano gli eventuali vincoli tra features. Il numero è pari a quello dei segmenti che compongono la triangolazione di Delaunay.

Per rendersi conto della grandezza del sistema utilizziamo ancora una volta l'esempio di un immagine con una risoluzione di 320x240 pixels. Il sistema senza vincoli di features ha 76.800 variabili e 153.840 vincoli, il che porta ad avere una matrice dei coefficienti con 11.814.912.000 elementi. Gestire matrici di tali dimensioni richiede un ottimizzazione del sistema molto curata, quale è quello di Matlab. Inoltre, un sistema di questo tipo sarebbe ingestibile se non fosse sparso. Per fortuna in questo caso lo è, infatti il numero di valori diversi da 0 in ogni riga è al massimo due. Gestendo le matrici sparse in modo opportuno è possibile superare almeno in parte l'ostacolo delle dimensioni.

CALCOLO DELLA SOLUZIONE DEL SISTEMA

Un sistema così grande richiede degli accorgimenti anche nella sua risoluzione. Matlab offre tutti gli strumenti per gestire matrici sparse. E' possibile lasciare al sistema la scelta del metodo di risoluzione migliore utilizzando l'operatore \, ma questo richiede del tempo ulteriore. La scelta migliore è sicuramente quella di richiamare direttamente la funzione che risolve con tecniche ricorsive nel senso dei minimi quadrati sistemi sparsi di grandi dimensioni, aiutando l'elaborazione con dei parametri che suggeriscono la forma attesa della soluzione.

```
x = lsqr(A,b,tol,maxIt,[],[],x0);
```

Nel commando riportato si chiede di trovare la soluzione x al sistema $Ax = b$. `tol` definisce la tolleranza che si decide di adottare; un valore pari a 10^{-6} garantisce un tempo di elaborazione tollerabile senza provocare eccessivi artefatti nell'immagine finale. `maxIt` limita il numero massimo di iterazioni che si vuole concedere al metodo di risoluzione iterativo. Infine, la variabile `x0` suggerisce l'effettiva forma attesa della soluzione. Questa matrice è stata inizializzata con i valori che si ottengono scalando l'immagine uniformemente senza considerare la funzione energia.

PRESTAZIONI

Un grande difetto di questo progetto è dovuto alle scarse prestazioni che riesce ad ottenere in termini di tempo di elaborazione. Come abbiamo visto il sistema da risolvere è estremamente grande. La sua stessa costruzione inizialmente necessitava di molto tempo. In seguito, come indicato nei precedenti paragrafi, si è capito come ridurre lo sforzo costruendo in pochi passi la struttura generale della matrice, senza dover considerare un vincolo per volta. In generale quindi il tempo di esecuzione dell'intera funzione di resize dipende esclusivamente dalla risoluzione del sistema. Questo ha anche impedito di eseguire test su immagini più grandi di 512x320 pixel, i quali già richiedono diversi minuti di tempo.

I tempi riportati in seguito fanno riferimento a test eseguiti su una macchina equipaggiata da un processore dual-core Intel Core 2 Duo a 2,4GHz e 2Gb di RAM DDR2 lavoranti ad una frequenza di 400MHz (800Mhz DDR).

L'immagine gattino.jpg in Figura 4 e in Figura 6 ha una risoluzione 300x225 e la riduzione ad una risoluzione di 150x225 richiede 1'55''. Di questi, 1'46'' sono spesi nella risoluzione del sistema, ovvero il 92% del tempo.

L'esempio in Figura 7 richiede 1'02'', di cui quasi 55s dovuti all'esecuzione della sola funzione `lsqr` di risoluzione del sistema, pari all'88% del tempo totale.

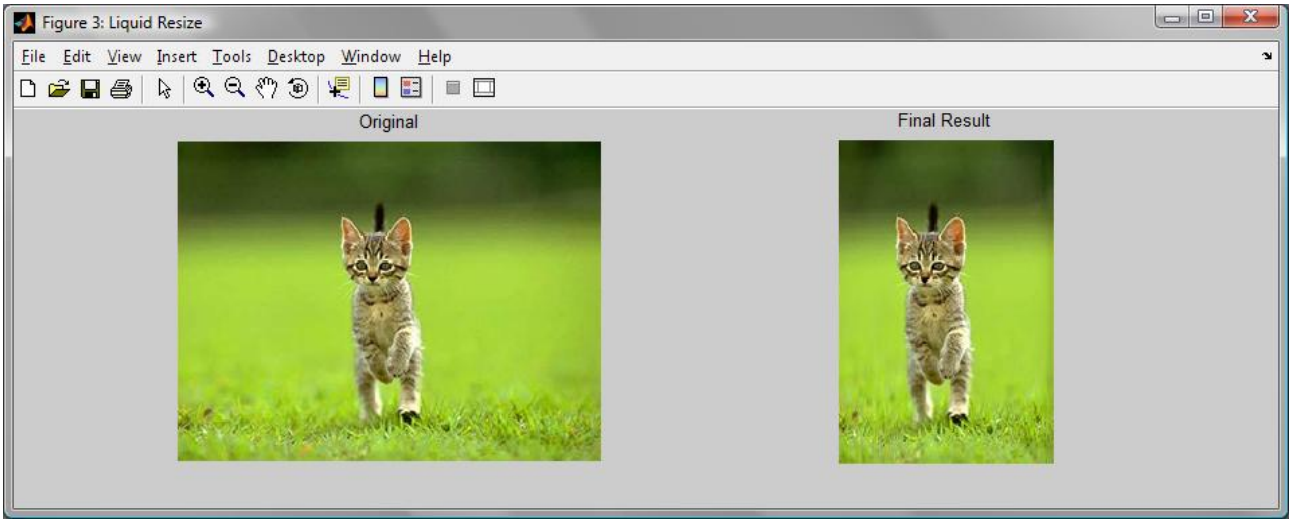

FIGURA 6 Primo esempio dei risultati ottenuti. L'immagine in questione è gattino.jpg e come è facile notare è stato ridotto orizzontalmente del 50%. L'immagine si presta molto bene all'esperimento, ma in questo modo è immediato apprezzare la fedeltà del gattino e la compressione dello sfondo, il tutto in modo completamente automatizzato.

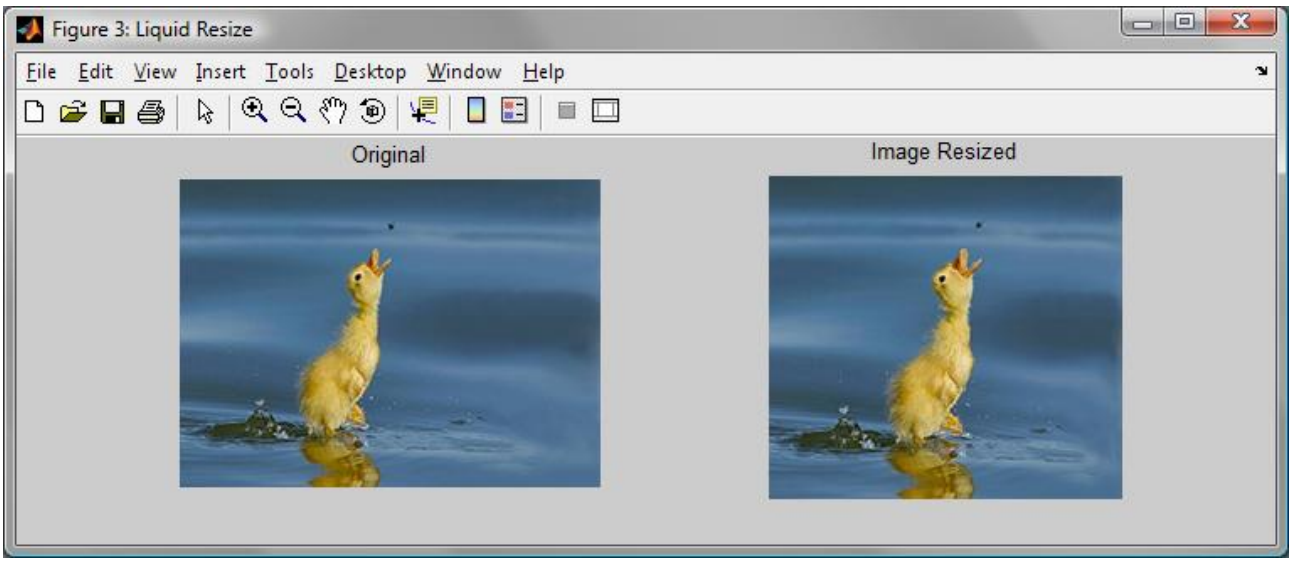

FIGURA 7 L'immagine in figura è un esempio dei risultati che si ottengono con l'esecuzione della nostra funzione di resize. Il file paperotto.jpg ha una risoluzione di 250x183 ed è stata ridimensionata a 200x183. Si può apprezzare come il papero sia rimasto intatto, così come l'increspatura dell'acqua. Il lato destro essendo molto omogeneo è stato compresso.

L'immagine in Figura 8, pur ammettendo la semplicità della prova, rappresenta la risoluzione massima a cui ci siamo spinti. L'immagine originale ha una risoluzione di 512x329. L'immagine ridimensionata è 427x329. Per l'operazione si sono resi necessari 14'14'' di cui 13'49' per la sola risoluzione del sistema. Questo significa che il 97% del tempo è stato speso nel calcolo delle nuove posizioni.

È quindi possibile osservare come all'aumentare delle dimensioni dell'immagine il rapporto tra il tempo di risoluzione del sistema e il tempo totale aumenti avvicinandosi velocemente al 100%. Questo perché chiaramente il numero di pixel cresce col quadraticamente con la lunghezza dei lati.

Và sottolineato come in questi test si limitino a ridurre solo una dimensione dell'immagine, nella fattispecie la larghezza. Intervenire anche sull'altezza comporterebbe un quasi raddoppio del tempo, perché il processo viene eseguito in egual modo due volte consecutive: la prima volta sull'immagine orientata come l'originale; la seconda volta sull'immagine trasposta o ruotata di 90°.

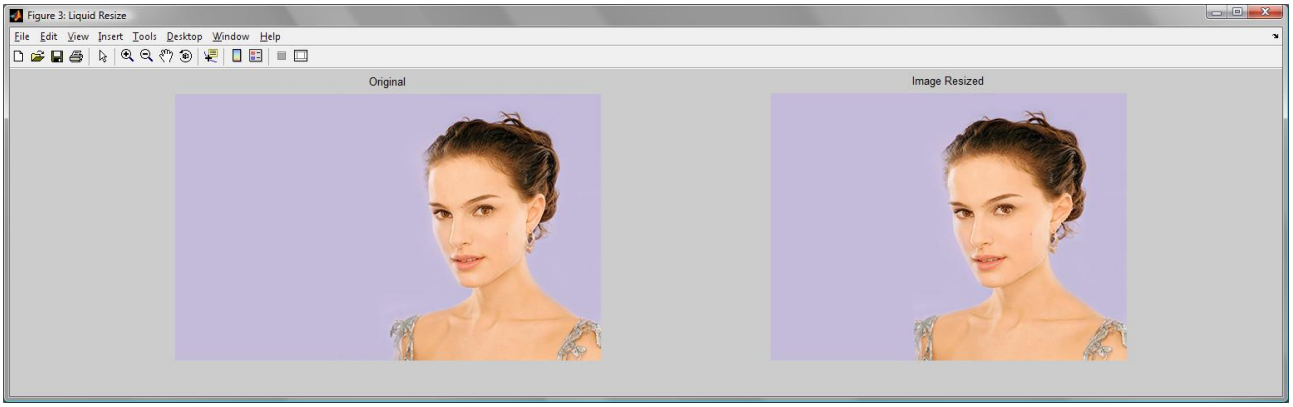

FIGURA 8 Questo file di nome natalie.jpg rappresenta il test più impegnativo che abbiato condotto. L'immagine originale ha una risoluzione di 512x329 pixel ed è stata ridotta a 427x329, per ottenere il formato televisivo di 4:3. Nell'insieme la ragazza risulta incredibilmente fedele all'originale, anche se analizzandola attentamente differisce in alcuni tratti difficilmente individuabili. Attenzione a non pensare che l'immagine sia stata solo tagliata. Dietro a questo risultato c'è stata un analisi, una elaborazione ed una decomposizione dell'immagine che è stata seguita da una ricomposizione.

Ottimizzazione del sistema

Come è stato mostrato le prestazioni in questo progetto sono poco soddisfacenti a causa delle dimensioni del sistema da risolvere. Per questo motivo si è cercato di trovare un modo per ottimizzare la sua risoluzione. Il sistema da gestire è lineare, sparso e sovra determinato. L'idea suggerita ancora una volta dal prof. Taddei è quella di convertire il sistema ad uno di tipo quadrato NxN in modo da ridurre le sue dimensioni. Questo comporta l'esecuzione di alcuni passaggi matematici.

Dato il sistema $\tilde{A}x = b$, per prima cosa occorre portalo in forma omogenea inserendo una variabile aggiuntiva w. Per cui risulta $\tilde{A}x - bw = 0$. Per tornare alla soluzione effettiva del sistema di partenza è sufficiente dividere tutto il vettore soluzione per w. Il primo passo quindi è quello di usare come nuova matrice dei coefficienti $A = [\tilde{A}| - b]$. Il nuovo sistema da risolvere è $Ax = 0$. Risolvere questo significa trovare il valore di x per cui si minimizza $\|Ax\|^2$, che equivale a $(Ax)^T(Ax)$ e di conseguenza $x^T A^T Ax$. Introducendo $K = A^T A$ è possibile eseguire la scomposizione SVD a valori singolari che porta $K = A^T A = U \cdot S \cdot U^T$ dove S è la matrice diagonale dei valori singolari. A questo è possibile definire $R = \sqrt{S} \cdot U^T$. Quella che si ottiene è una matrice quadrata i cui valori di x che minimizzano $\|Rx\|^2$ sono gli stessi del sistema omogeneo di partenza $Ax = 0$.

I passaggi descritti consentono di ricavare un sistema NxN che può essere risolto più efficacemente. Va detto tuttavia che anche il codice per ottenere la matrice R è pesante, e per questo motivo quest'ottimizzazione non è stata mantenuta nella versione finale del progetto.

Ricostruzione dell'Immagine

Per ottenere l'immagine finale ridimensionata occorre interpretare la soluzione ottenuta dalla risoluzione del sistema lineare. Questa infatti riporta la nuova posizione che ogni pixel deve occupare nell'immagine da sintetizzare, ma occorre definire per ogni nuovo pixel il colore dei tre canali RGB.

Per prima cosa si riporta il vettore soluzione ad una forma matriciale più comoda, in modo che sia facile osservare la distribuzione dei pixel:

$$\begin{bmatrix} 0{,}9 \\ 1{,}3 \\ 2{,}1 \\ 1{,}1 \\ 1{,}4 \\ 1{,}9 \end{bmatrix} \rightarrow \begin{bmatrix} 0{,}9 & 1{,}3 & 2{,}1 \\ 1{,}1 & 1{,}4 & 1{,}9 \end{bmatrix}$$

L'esempio riportato riguarda un immagine sorgente 3x2 che viene ridotta a 2x2. Come vediamo i valori ottenuti dal sistema sono continui e non più discreti. Inoltre il valore dell'ultima colonna si avvicina a quello della nuova larghezza dell'immagine (nell'esempio pari 2). Ora è necessario tornare al discreto delimitando intervalli di larghezza 1 a cui i pixel possono appartenere. Nel progetto si è scelto di iniziare dal'intervallo $[0{,}5\ 1{,}5)$ seguito da $[1{,}5\ 2{,}5)$, ecc... . Questo significa che i primi due pixel dell'esempio in posizione 0,9 e 1,3 appartengono al primo pixel. Il terzo pixel in posizione 2,1 invece verrà rimappato nel secondo ed ultimo pixel della nuova immagine. Ovviamente più pixel dell'immagine di partenza di troveranno a condividere lo stesso pixel nell'immagine finale. Per fondere più pixel assieme si è semplicemente considerata la media aritmetica dei 3 canali RGB separatamente. L'operazione è piuttosto rapida considerando che le posizioni dei pixel sono monotone crescenti.

Pixel Orfani

L'immagine ottenuta dopo la ricostruzione descritta nel paragrafo precedente potrebbe già considerarsi definitiva. Purtroppo però test empirici hanno mostrato come sia frequente che alcuni pixel dell'immagine finale non abbiano un corrispondente nell'immagine sorgente. Questo accade quando l'intervallo per un pixel non è soddisfatto da nessuna posizione calcolata dal sistema. In teoria questo non dovrebbe accadere, perché la distanza massima concessa tra due pixel è 1 e l'immagine viene sempre compressa. Tuttavia, la soluzione del sistema è affetta da errori di calcolo numerico per cui, troncamenti e arrotondamenti dei valori uniti a tecniche di risoluzione iterative con tolleranza limitata portano al verificarsi di questo fenomeno. Va comunque sottolineato che a meno di problemi di altra natura, i pixel orfani non rappresentano un grosso problema. Il loro numero è assai ridotto e spesso appaiono solo ai lati dell'immagine. Sebbene quindi questo sia un inconveniente di poco conto, è necessario gestirlo.

La funzione di ricostruzione dell'immagine quando trova un pixel orfano marca la sua posizione in un apposita matrice che verrà utilizzata subito dopo per l'eliminazione degli orfani. Infatti, per ogni pixel orfano individuato viene assegnato un colore copiato direttamente dai pixel adiacenti. Il metodo scorre l'immagine riga per riga dall'alto al basso e da sinistra verso destra. I colori vengono assegnati in base a tre casi distinti: se il pixel si trova sul bordo sinistro il suo colore viene copiato dal primo pixel non orfano alla sua destra; allo stesso modo, se il pixel si trova sul bordo destro il suo valore sarà quello del pixel direttamente alla sua sinistra; infine, se il pixel è in una posizione centrale assumerà come colore la media tra il pixel alla sua sinistra e il primo pixel non orfano alla sua sinistra. Questo procedimento garantisce un eliminazione totale degli orfani senza danneggiare l'immagine.

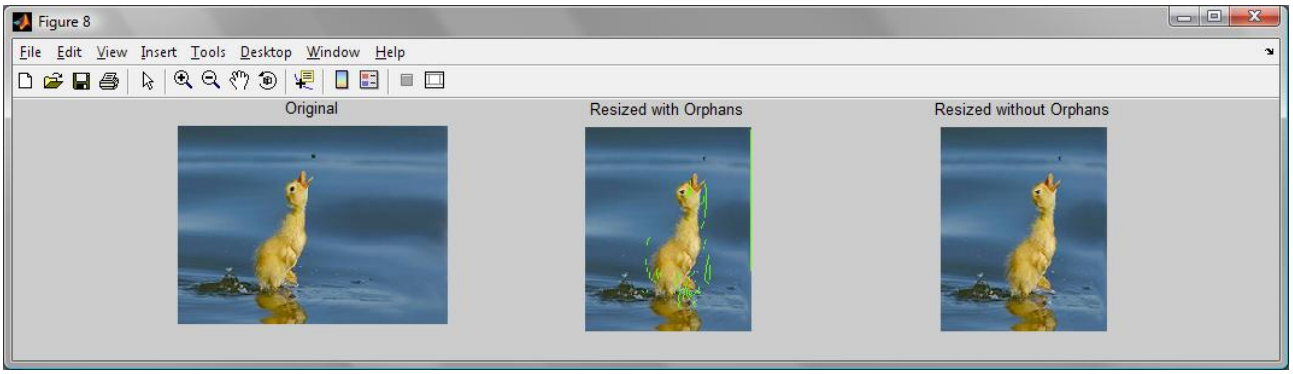

FIGURA 9 Partendo da sinistra vengono mostrate l'immagine originale, l'immagine ridimensionata con orfani in verde e l'immagine finale dopo l'eliminazione degli orfani. Notare come il bordo sinistro sia affetto da orfani e come quelli interni siano stati rimossi senza pregiudicare l'immagine.

FUNZIONE DI RESIZE

Per poter analizzare meglio il comportamento dell'algoritmo nel suo complesso si è deciso di implementare un metodo che descrive la funzione di resize. Con questo termine si intende uno schema che mostra come i pixel sono stati spostati dall'immagine originale a quella ridimensionata. La matrice con le posizioni calcolate non consente di valutare con immediatezza in che modo è stata modificata l'immagine. Nel nostro progetto viene quindi sintetizzata un'ulteriore immagine in scala di grigi in cui i colori scuri sono associati a grosse deformazioni e colori chiari ad una fedeltà con l'originale. La Figura 11 mostra un esempio di funzione di resize. Per ottenere questo risultato è stato sufficiente controllare la distanza tra i pixel. Se inizialmente i pixel sono tutti a distanza 1, nell'immagine ridimensionata troveremo valori compresi tra 0 e 1. Se la distanza è vicino a 1 allora quella parte di immagine è stata mantenuta fedele all'originale. Se il valore è più vicino a 0 allora quella zona è stata compressa per rientrare nel nuovo spazio concesso.

Se questo metodo è seguito per il ridimensionamento sia orizzontale che verticale è possibile fondere insieme i risultati per ottenere una funzione totale di resize. Si possono molti modi di fusione. Si può scegliere di considerare il minimo, la somma o la radice dei quadrati. Nell'esempio in Figura 11 è stata utilizzata la somma.

Va sottolineato come la seconda fase di resize verticale lavora su un immagine che è già stata scalata orizzontalmente e che è quindi più piccola. Per creare la funzione totale di resize occorre prima estendere lo schema in modo che torni alla dimensione originale. I dettagli di questa operazione sono descritti meglio nel codice sorgente.

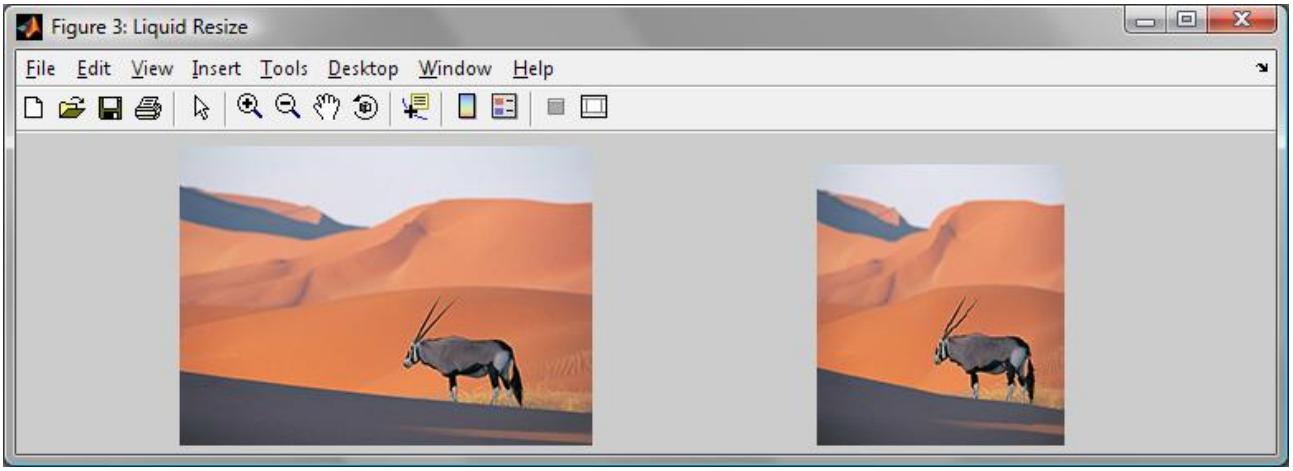

FIGURA 10 Esempio di ridimensionamento sia orizzontale che verticale del file antilopeMedium.jpg. L'immagine ha una risoluzione di partenza di 250x188 pixels e viene portata ad una risoluzione di 150x170.

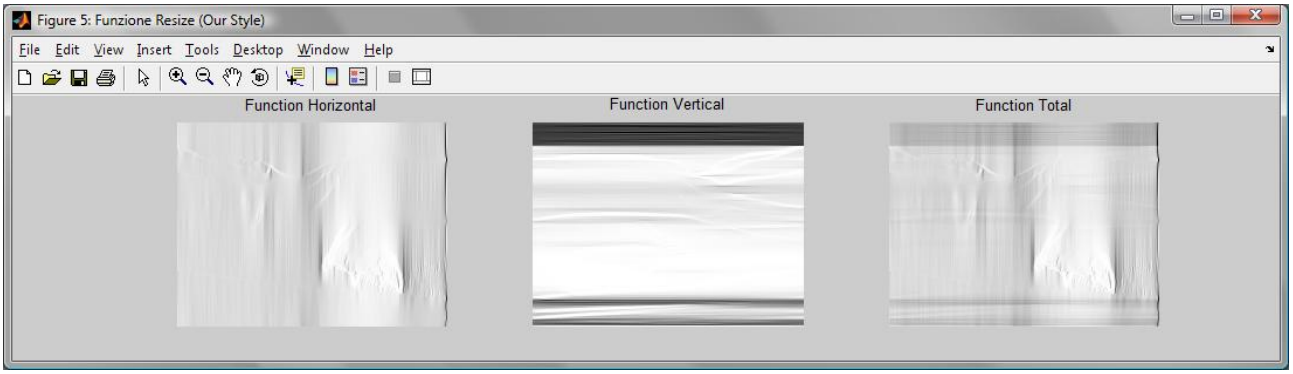

FIGURA 11 Le tre immagini mostrano un esempio di funzione energia ottenute dall'immagine di Figura 10. La prima a sinistra si riferisce al ridimensionamento orizzontale. La parte più chiara è quella che comprende l'antilope a l'intersezione delle dune in alto. L'immagine centrale riguarda invece il resize verticale ed è molto evidente come la parte alta e la parte bassa siano state ridotte moltissimo rispetto al resto dell'imamgine. Ancora una volta la parte che è stata conservata meglio è quella che interessa l'antilope. L'ultima immagine a destra compone le due funzioni.

COME UTILIZZARE LA FUNZIONE LIQUIDRESIZE

Il progetto di concretizza in due funzioni Matlab sostanzialmente simili. Una consente di modificare sia la larghezza che l'altezza, mentre l'altra solo la larghezza. Inoltre, mentre la prima versione modifica la funzione energia per inserire le informazioni sulle features, la seconda aggiunge i vincoli direttamente nel sistema. I dettagli di queste operazioni sono stati forniti nei capitoli precedenti.

La funzione si lancia con:

```
resized_img = liquidResize(img,new_Width,new_Height);

resized_img = liquidResize(img,new_Width);
```

CONCLUSIONI

OSSERVAZIONI E PROBLEMI RISCONTRATI

La versione definitiva del progetto si comporta bene in diverse situazioni ma, soprattutto nella versione con i vincoli delle features inseriti direttamente nel sistema, può portare a strani comportamenti dovuti probabilmente ad errori nella risoluzione del sistema. E' immediato notare una netta riduzione di pixel orfani e di artefatti lasciando più tempo alla risoluzione del sistema, che però impiega già moltissimo tempo come è stato mostrato nel capitolo relativo alle prestazioni. Purtroppo, esistono anche molti casi in cui la funzione energia così come viene calcolata non consente un resize ragionevole portando a grosse deformazioni. Questo avviene quando i soggetti nelle immagini hanno zone a basso constrato, e lo sfondo, che dovrebbe essere un elemento secondario, ha invece un forte contrasto. Oppure si nota molto la deformazione di segmenti rettilinei che nella versione finale non risultano più tali. L'immagine seguente mostra un esempio di questa situazione:

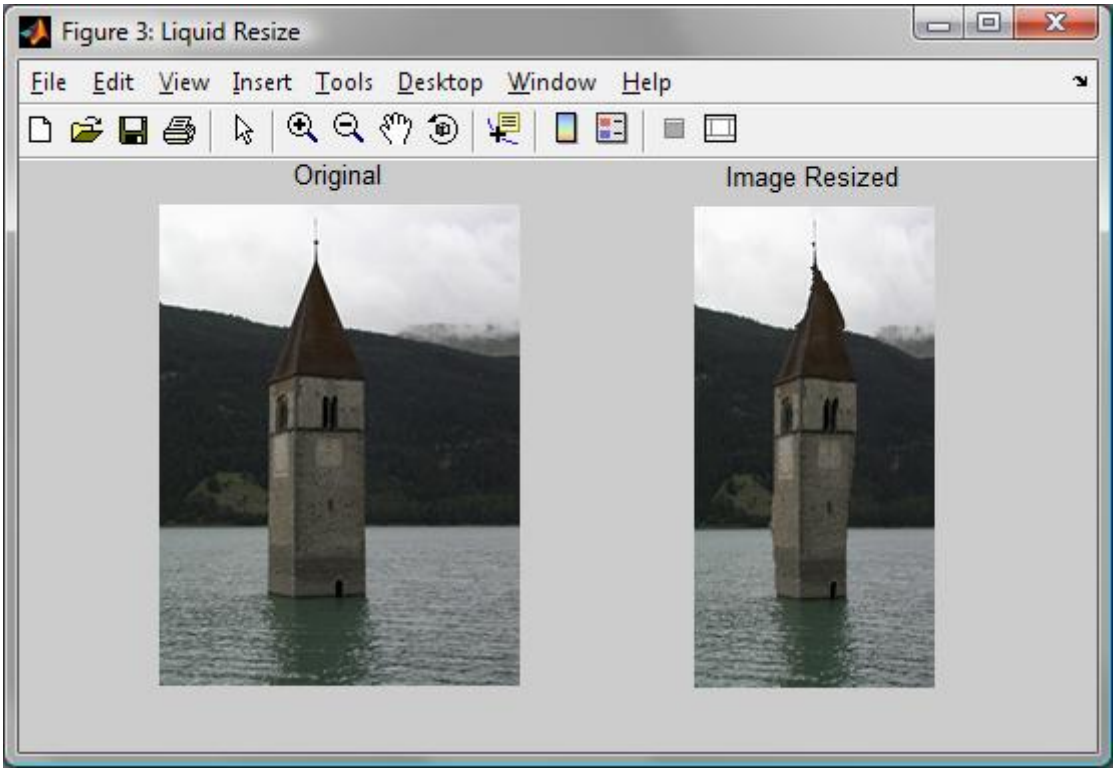

FIGURA 12 In immagini come questa in cui sono presenti strutture o edifici è facile notare delle deformazioni in quanto segmenti rettilinei risultano curvati da una compressione non uniforme.

La ricerca delle features aiuta a mantenere le proporzioni dei soggetti dell'immagine, ma se ne sono presenti moltissime il loro utilizzo viene vanificato. Il tutto peggiora senza una vera funzione di segmentazione.

Infine, in immagini con sfondi naturalistici e comunque con molti dettagli, la funzione di resize tende a diventare uniforme, perché non riesce a trovare zone da comprimere più di altre. Questo è più evidente inserendo i vincoli nella funzione energia perché la saliency presenta un reticolo che copre l'intera immagine.

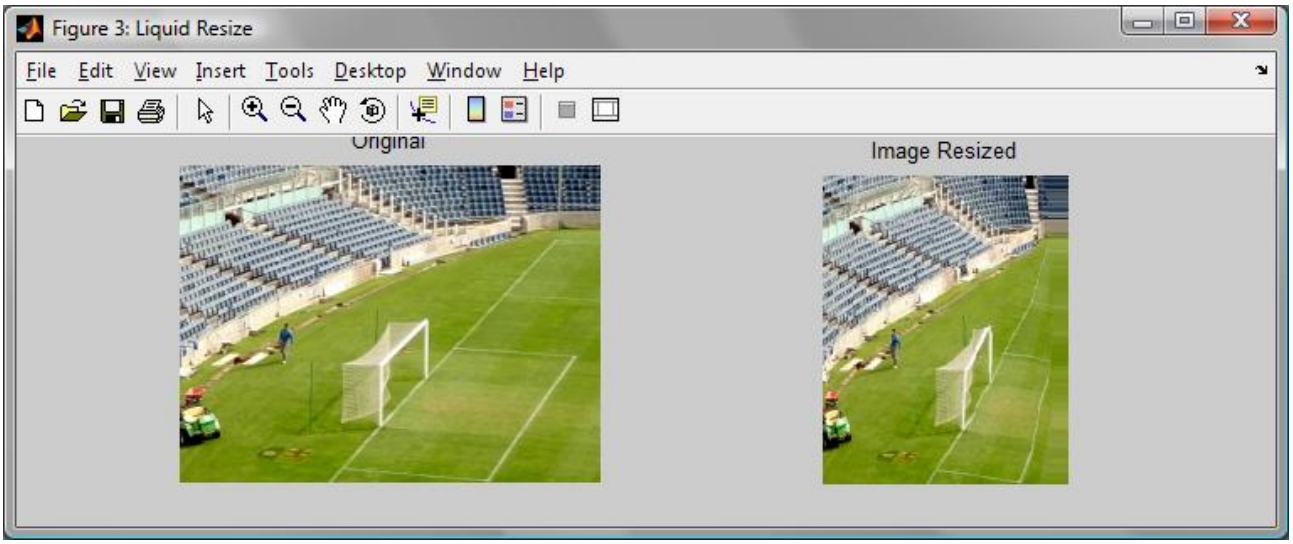

FIGURA 13 Un altro esempio di problemi riscontrati. Questa immagine presenta moltissime features che coprono in larghezza tutta l'immagine. Mantenere invariate le proporzioni significa comprimere l'immagine in modo uniforme. Inoltre, anche in questo caso i tratti rettilinei vengono curvati. Ultimo problema riscontrabile in quest'immagine è un intensa presenza di pixel orfani sul lato destro.

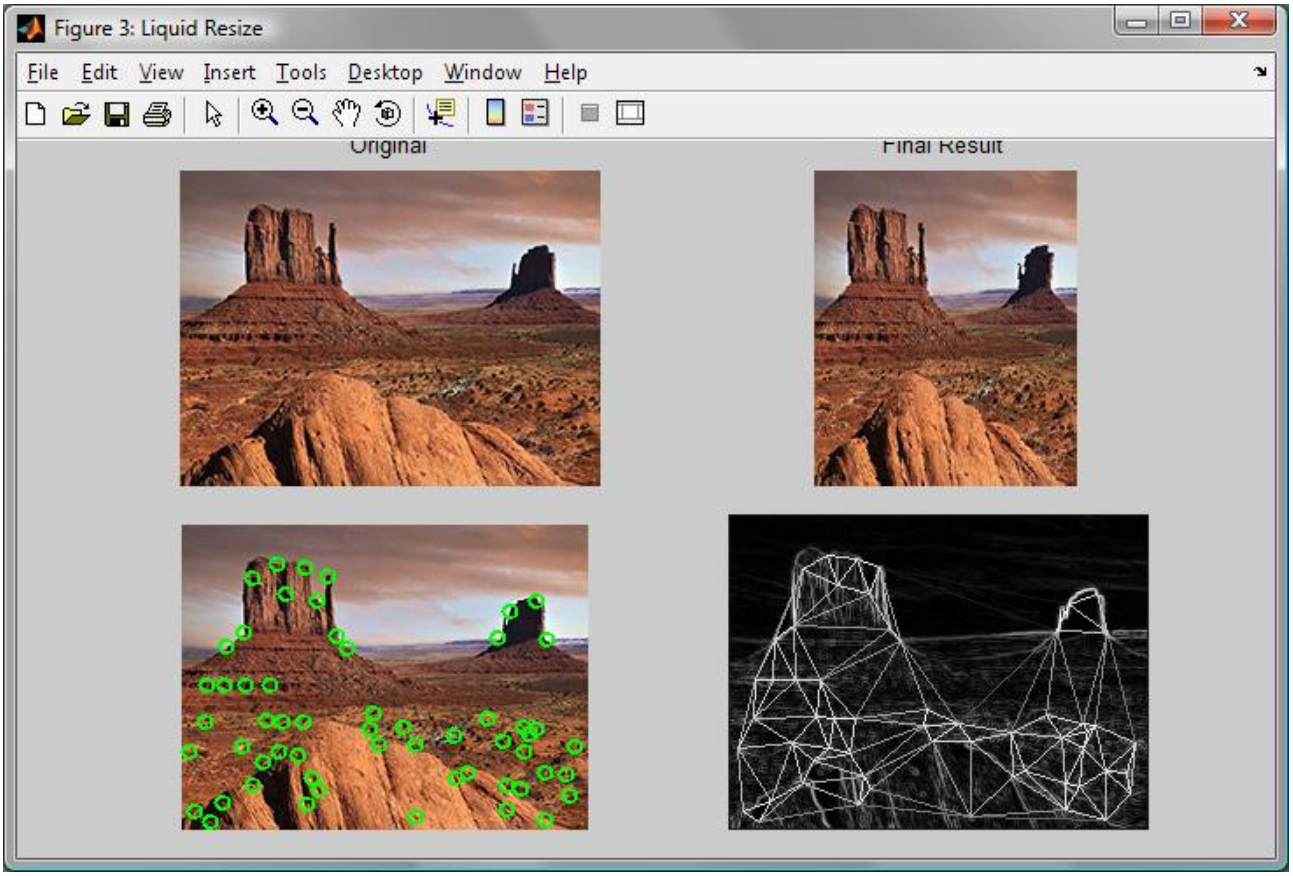

FIGURA 14 Nei paesaggi naturalistici è facile trovare molte features diffuse in tutta l'immagine, il che porta a creare un reticolo troppo esteso ed in definitiva inutile. Il risultato finale è una compressione uniforme classica.