UNIVERSITA' DEGLI STUDI DI MILANO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica

TESI DI LAUREA

RESTAURO DI IMMAGINI DIGITALI TRAMITE TECNICHE DI INPAINTING

Relatore: Chiar.mo Prof. Ing. Alberto N. BORGHESE

Correlatore: Dott. Iuri FROSIO

Candidato:

Guido CRIVELLARI

Anno Accademico 2005 – 2006

Ai miei genitori,
per tutto ciò che ho da loro imparato e ricevuto.
A mio papà,
per i sacrifici che ha sempre fatto per permettermi di arrivare fino a questo punto e per il bene che da sempre mi ha dimostrato.
A mia mamma,
sono sicuro che ne saresti orgogliosa e fiera, come lo sei sempre stata di me.

1. Introduzione

"*Fu attraverso il confronto con la fotografia che l'arte andò via via distaccandosi, per differenziarsi, dal concetto classico della mimesi, e si costituì in proprio una morfologia e un lessico senza radici naturalistiche. Ma la divisione di campo non durò, la fotografia invase anche quel dominio: si presentò come operazione più mentale che tecnica, potenzialmente creativa come e più dell'arte*". *(C. G. Argan, 1989)*

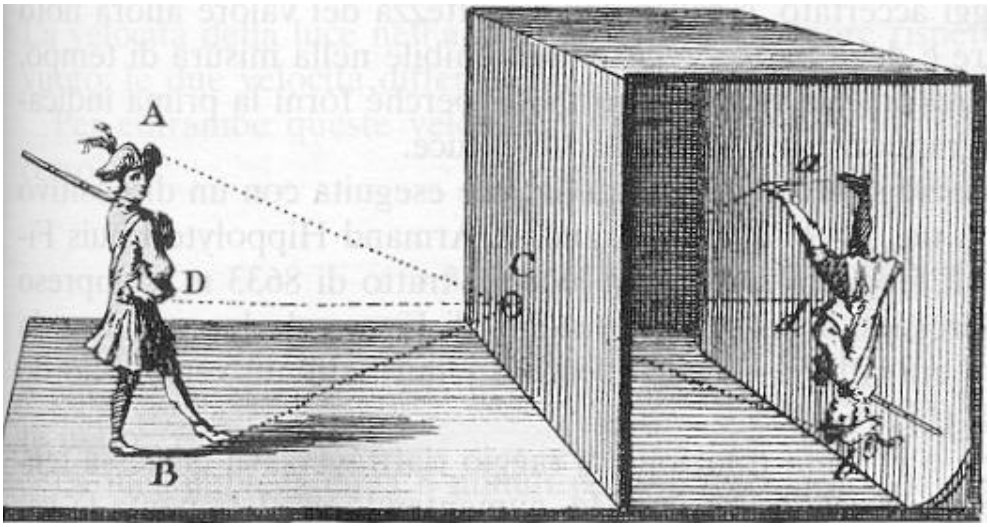
Sin dall'alba dei tempi l'uomo ha sentito il bisogno di lasciare testimonianza di sè, dell'ambiente che lo circonda, delle proprie emozioni nel vivere quotidiano. Grazie a questa sua singolare esigenza oggi possiamo ammirare opere come i grafitti dell'uomo primitivo, gli affreschi rinascimentali, le tele degli impressionisti e via discorrendo. Insomma, tramandare la propria immagine ai posteri è da sempre desiderio dell'uomo, indipendentemente dal ceto sociale di appartenenza.
Purtroppo, fino al XIX secolo, la pittura fu prerogativa di pochi. I pittori erano troppo costosi perché contadini e semplici braccianti potessero permettersi un ritratto o un dipinto che li riprendesse nella loro umile dimora.
I cambiamenti nacquero proprio grazie all'invenzione della fotografia.
Ora anche i poveri potevano pagare i pochi centesimi necessari per una "carte de visite", come erano chiamati i ritratti ottocenteschi formato cartolina. E proprio dal ritratto la fotografia ebbe il primo fondamentale impulso. Chiunque, anche chi non aveva l'abilità del pittore, poteva improvvisarsi ritrattista.
Già dieci anni dopo la nascita ufficiale della fotografia, gli studi fotografici specializzati nel ritratto potevano contarsi a centinaia, sparsi per tutta l'Europa.
Il ritratto divenne oggetto di consumo popolare; davanti alla macchina fotografica passano e passavano tutti, almeno una volta nella vita: in occasione delle nozze o del servizio militare.

Fin dagli inizi il ritratto fotografico ripete atteggiamenti e illuminazione del ritratto pittorico. La classica ripresa fotografica “a mezzo busto” è una imitazione di quella del ritratto pittorico. L’atteggiamento della persona imita ancora oggi, quando non ce ne sarebbe bisogno, le pose immobili che il soggetto era costretto ad assumere durante le lunghe sedute davanti al pittore. La raccomandazione del pittore al suo soggetto di rimanere fermo è passata alla fotografia.
Anche quando riprende con un tempo di scatto inferiore al millesimo di secondo, più che sufficiente a fermare un piccolo movimento, il fotografo raccomanda al soggetto di rimanere fermo.
L’illuminazione usata dal fotografo, infine, ricalca quella dei pittori.
I primi fotografi si formarono nelle scuole di pittura e in particolare del ritratto in miniatura. Non è certo un caso che il dagherrotipo, la prima immagine eseguita a macchina, avesse dimensioni e aspetto della miniatura.
Quella che si potrebbe definire democratizzazione del ritratto non ha interessato solo il soggetto, con il passare del tempo ha riguardato anche lo stesso fotografo. Oggi tutti, non importa che si possegga una fotocamera costosa, piuttosto che una semplice compatta, possono fare buoni ritratti.
Purtroppo per noi, le prime fotografie venivano realizzate su materiali alquanto delicati e facili all’usura, così, dopo soli cento anni, molto del materiale realizzato dai primi fotografi, era andato perso a causa della cattiva conservazione.
Un tempo inesorabilmente breve se pensiamo ai rinvenimenti millenari di geroglifici o ai più recenti affreschi risalenti a secoli successivi alla morte di Cristo.
Ma non tutto il materiale fotografico fu perso. Proprio per questo motivo un ramo specializzato di restauratori focalizzò la propria attenzione sul recupero di queste fotografie danneggiate.
Oggi siamo ormai nell’era del digitale e anche la fotografia è entrata in questo “magico” mondo, dove l’usura del tempo non esiste, dove ogni immagine può essere trasformata in una sequenza di bit e resa immortale, eterna.

Grazie a questa possibilità si è oggi in grado di acquisire le fotografie danneggiate ed elaborarle in modo da ottenere risultati migliori di quelli permessi dal restauro manuale, grazie all'utilizzo di appositi algoritmi matematici proposti dai ricercatori.
Si parla allora di restauro digitale, che sarà proprio l'argomento di questa tesi.
Vediamo ora una breve carrellata sulla storia della fotografia, la sua scoperta e la sua evoluzione, fino ai giorni nostri.

2. Storia della fotografia

La parola “fotografia” ha origine da due parole greche photos, luce, e grafia, scrittura. Letteralmente quindi fotografia significa scrivere con la luce.
E’ un’invenzione con “molti padri”, ebbe infatti origine dalla convergenza dei risultati ottenuti da numerosi sperimentatori, sia nel campo dell’ottica, con lo sviluppo della camera oscura, sia in quello della chimica, con lo studio delle sostanze fotosensibili.
La prima camera oscura fu realizzata molto tempo prima che si trovassero dei procedimenti per fissare con mezzi chimici l’immagine ottica da essa prodotta, sembra che il primo ad averla concepita sia stato Aristotele, addirittura nel IV secolo a.C., allo scopo di osservare un’eclissi di sole.
La camera oscura essenzialmente consiste in una scatola completamente chiusa, ma con un piccolissimo foro su una delle pareti. Se poniamo la scatola con il foro rivolto verso un oggetto illuminato, sulla superficie interna opposta al foro verrà proiettata l’immagine dell’oggetto stesso, ma molto più piccola e capovolta.



2.1 – Esempio di camera oscura

La camera oscura venne utilizzata soprattutto dagli artisti del Rinascimento per proiettare, su pareti o su tele, immagini che servivano da falsariga per realizzare un disegno o un dipinto. Si sa ad esempio che Raffaello ne fece ampio uso e con lui tutti quegli artisti che avevano necessità di riprodurre ampie prospettive con un fedele disegno dei paesaggi.

Joseph Nicéphore Niepce

Le prime occupazioni della camera oscura per la fotografia si ebbero con J. N. Niepce, al quale viene abitualmente attribuita l'invenzione della fotografia.
Nel 1813 egli iniziò a studiare i possibili perfezionamenti da apportare alle tecniche litografiche e da queste ricerche sviluppò un interesse per la registrazione diretta di immagini sulla lastra litografica, senza l'intervento dell'incisore.
In collaborazione con il fratello Claude, Niepce cominciò a studiare la sensibilità alla luce del cloruro d'argento e, nel 1816, ottenne la sua prima immagine fotografica, utilizzando un foglio di carta sensibilizzato con, probabilmente, cloruro d'argento.
L'immagine, tuttavia, non poté essere fissata completamente, per cui Niepce fu indotto a studiare la sensibilità alla luce di numerose altre sostanze, soffermandosi sul bitume di Giudea che possiede la proprietà di divenire insolubile in olio di lavanda in seguito a esposizione alla luce.
La più antica immagine oggi esistente è una di quelle che Niepce ottenne nel 1824, utilizzando una camera oscura nella quale l'obiettivo era una lente biconvessa, dotata di diaframma e di un rudimentale sistema di messa a fuoco.

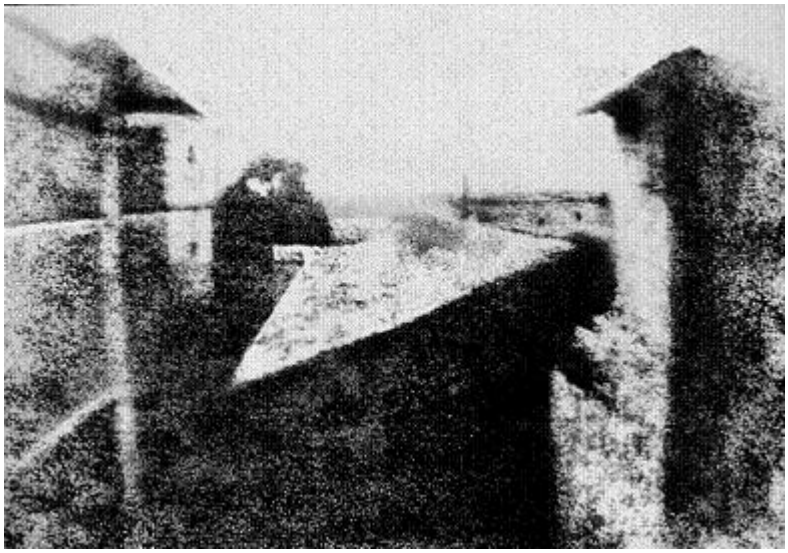


Fig. 2.2 - Vista della camera a Le Gras, J.N.Niepce 1826

Louis-Jacques-Mandé Daguerre e la dagherrotipia

Daguerre era un artista e chimico basco riconosciuto come il padre del processo fotografico chiamato dagherrotipo.

Nel 1829 fondò con Niepce una società per lo sviluppo delle tecniche fotografiche.

Nel 1839 il fisico Francois Arago descrisse all'Accademia delle Scienze di Parigi un procedimento messo a punto da Daguerre, il dagherrotipo.

Questo si ottiene utilizzando una lastra di rame su cui è stato applicato elettroliticamente uno strato d'argento, quest'ultimo viene sensibilizzato alla luce con vapori di iodio. La lastra deve quindi essere esposta entro un'ora e per un periodo variabile tra i 10 e i 15 minuti.

Lo sviluppo avviene mediante vapori di mercurio a circa 60°C, che rendono biancastre le zone precedentemente esposte alla luce. Il fissaggio conclusivo si

ottiene con una soluzione di iposolfito di sodio, che elimina gli ultimi residui di ioduro d'argento.
L'immagine ottenuta, il dagherrotipo, non è riproducibile e deve essere osservata sotto un angolo particolare per riflettere la luce in modo opportuno. Inoltre, a causa del rapido annerimento dell'argento e della fragilità della lastra, il dagherrotipo veniva racchiuso sotto vetro, all'interno di un cofanetto impreziosito da eleganti intarsi in ottone, pelle e velluto, volti anche a sottolineare il valore dell'oggetto e del soggetto raffigurato.

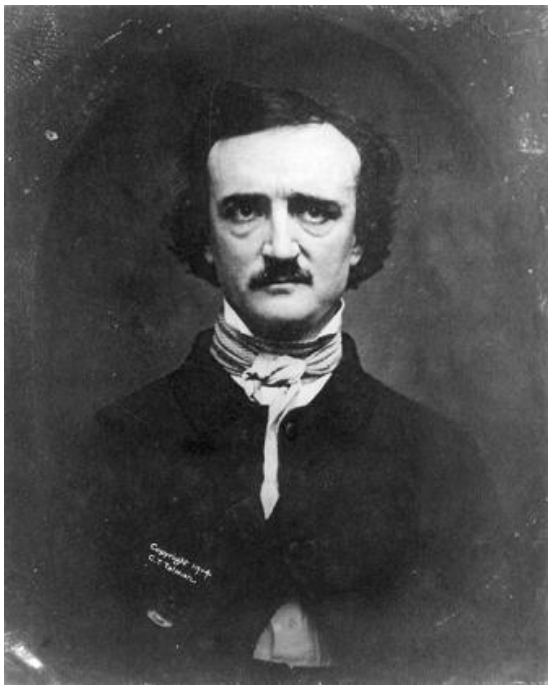


Fig. 2.3 - Famoso dagherrotipo di Edgar Allan Poe del 1848

William Fox Talbot e la calotipia

La notizia dell'invenzione esposta da Arago suscitò l'interesse di Talbot, che dal 1835 sperimentava un procedimento fotografico denominato calotipia.
La calotipia permette di produrre copie di un'immagine utilizzando il negativo, di cui Talbot fu l'inventore.
La qualità della stampa risulta però inferiore rispetto al dagherrotipo, specialmente nei dettagli. Inoltre la possibiltità di ottenere immagini riproducibili, non rendeva il prodotto calotopico prezioso come l' opera unica del dagherrotipo.

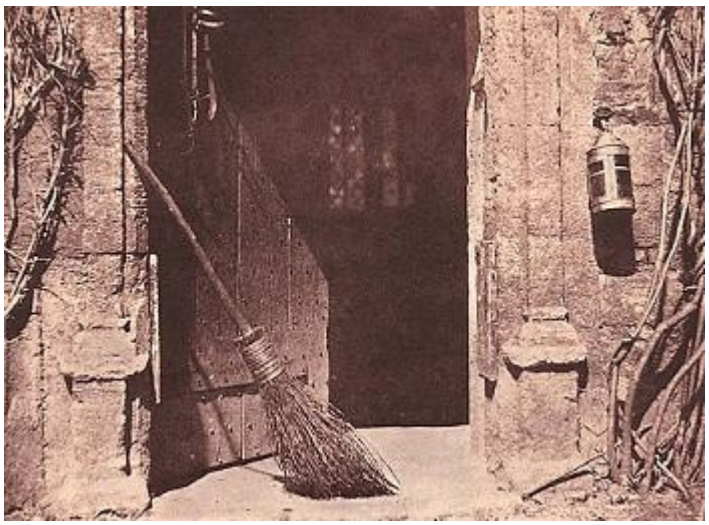


Fig. 2.4 - Calotipia proveniente dal libro "Pencil of Nature", 1844

Ambrotipia

L’ambrotipia è un procedimento fotografico per lo sviluppo di immagini su vetro inventato nel 1852 da Frederick Scott Archer.

A differenza del dagherrotipo, la visione dell'ambrotipo avveniva senza la necessità di inclinare la lastra, comunque a causa del ridotto contrasto causato dall'assenza di bianchi puri, che venivano realizzati in gradazioni di grigio, era necessaria una adeguata fonte di luce. L'ambrotipo, essenzialmente un negativo su vetro, spianò la strada alla stampa di fotografie su carta in una qualità superiore a quella ottenuta dalla calotipia. Il procedimento piuttosto economico ne permise una rapida diffusione e un utilizzo protratto agli inizi del XX secolo.

Fig. 2.5 - Ambrotipia del 1860

Ferrotipia

La ferrotipia (o tintype) è un procedimento fotografico per lo sviluppo di immagini non riproducibili su una lastra di metallo. Inventata nel 1852 dall'americano Adolphe Alexandre Martin, fu perfezionata da Hannibal Smith nel 1856, che utilizzò dell'acciaio smaltato al posto della latta del processo originale.
La ferrotipia soffriva di problemi ereditati dal supporto metallico su cui l'immagine era impressa. Il tempo arruginisce il metallo compromettendo l'immagine, così come la flessibilità della lastra provocava distaccamenti del materiale sensibilizzato.
La qualità era equiparabile a quella dell'ambrotipia, quindi contrasto ridotto e poca nitidezza, specialmente in confronto con la dagherrotipia.

Fig. 2.6 - Ferrotipo di un afro-americano, 1870 circa

Richard Leach Maddox e Hermann Vogel

Nel 1871 Maddox scoprì un nuovo procedimento chimico per la creazione di emulsioni utilizzando gelatina animale e bromuro d'argento.
Con questa tecnica la gelatina agisce da legante dei sali d'argento e, quando incorporata al bromuro di cadmio e al nitrato d'argento, forma un'emulsione.
I primi materiali fotografici erano sensibili solo alla luce blu, quindi il rosso e il verde erano restituiti con grigi molto più scuri del blu.
Nel 1873 Hermann Vogel, professore di chimica di Berlino, dimostrò che era possibile sensibilizzare le lastre anche agli altri colori, trattando le emulsioni con dei coloranti.

George Eastman e la Kodak camera

Il vero salto di qualità fu compiuto da George Eastman, che nel 1888 inventò la prima Kodak camera (chiamata così per assonanza con il rumore prodotto da una macchina fotografica) , un apparecchio di piccole dimensioni che conteneva un rullo di carta speciale (la futura pellicola), per 100 pose. I fotografi che se ne servivano non dovevano più curarsi dell'uso della camera oscura o degli agenti chimici; una volta finiti gli scatti, bastava infatti portare alla Kodak la macchina fotografica e ritirala dopo qualche giorno, di nuovo pronta all'uso insieme alle fotografie già sviluppate. Fu una vera e propria rivoluzione che si può riassumere con lo slogan che Eastman aveva coniato per rappresentare la sua invenzione : "Tu premi il bottone e noi facciamo il resto". Nasceva la fotografia istantanea.

Fig. 2.7 - Kodak camera, 1890

Ferdinand Hurter e Charles Driffield

Nel 1890 Hurter e Driffield diedero inizio a complessi studi e approfondimenti tecnico-scientifici sulle emulsioni e sulla sensibilità delle stesse, che originarono la nuova dottrina denominata sensitometria, ovvero quella parte dell'ottica fotografica attinente lo studio della relazione che intercorre tra esposizione e annerimento di un'emulsione fotografica.

Taylor

Nel 1893 l'inglese H. D. Taylor progettava, per la Cooke & Sons, un obiettivo del tutto nuovo, che avrebbe annoverato fra i suoi discendenti molti nobili rampolli: un elemento positivo e uno negativo cementati insieme, chiamato Tripletto di Cooke. L'idea di partenza era costituita dal doppietto acromatico di Chevalier. Quest'ultimo era anastigmatico e con tre sole lenti non collate.

Fu poi perfezionato nel 1902 da P. Rudolph, con l'introduzione di un elemento posteriore collato.

Le Reflex e il processo Polaroid

Altri progressi si ebbero con l'introduzione del sistema Reflex nel 1928, che utilizzava degli strati anti-riflesso sulle superfici esterne delle lenti che migliorarono enormemente la trasmissione tra aria e vetro e il contrasto degli obiettivi.

Polaroid è il nome di uno speciale foglio di plastica utilizzato per polarizzare la luce. Il brevetto, registrato nel 1929 e sviluppato successivamente nel 1932 da Edwin H. Land, consisteva in un particolare procedimento con il quale si poteva ottenere istantaneamente una copia positivia della fotografia scattata, direttamente dalla macchina fotografica.

La fotografia digitale

Con gli anni settanta è iniziata l'epoca degli automatismi e dell'elettronica applicata alla fotografia. Si iniziarono a studiare apparecchi fotografici in grado di trasformare l'immagine ottica in una distribuzione di cariche elettriche su silicio drogato, tale da essere tradotta, poi, in un segnale video.

L'immagine così ottenuta è immagazzibile su supporti di memoria di massa ed è quindi maneggiabile tramite qualsiasi moderno personal computer.

Proprio quest'ultima caratteristica ha permesso l'aumento esponenziale degli appassionati alla fotografia. Basta avere un PC, una macchina fotografica digitale e uno dei sempre più diffusi programmi di grafica, per ottenere risultati davvero sorprendenti partendo dalle proprie stampe.

3. Classificazione dei difetti

Una stampa fotografica è ciò che usualmente viene indicato con il nome di *positivo* di un'immagine. E' un foglio di carta con una mistura di gelatina e nitrato d'argento sulla sua superficie. L'immagine è creata attraverso la sensibilità del nitrato d'argento alla luce.
Gli esperti suggeriscono di conservare le fotografie in condizioni di temperatura e umidità controllate, ma di questo daremo un breve accenno succesivamente. Inoltre, la prolungata esposizione alla luce solare dovrebbe essere evitata, dato che può modificare i colori fino a creare un effetto "scolorimento" sulla superficie della fotografia.

Fig. 3.1 - Effetto scolorito dovuto alla prolungata esposizione alla luce

Come già accennato, agli albori della fotografia, la non curanza nella conservazione del materiale fotografico era all'ordine del giorno. Il risultato consiste in un'infinità di reperti in pessimo stato conservativo.
I difetti delle vecchie stampe fotografiche possono essere divisi in classi, in accordo con l'origine del danno.

Possiamo avere: *danni meccanici*, *danni chimic*i e *danni dovuto a deposito*.

Danni meccanici

Questa categoria contiene tutti i difetti che sono originati dall'incuratezza nel maneggiamento del materiale fotografico.
Più in dettaglio abbiamo:

- Le crepe, pesanti deterioramenti dell'aspetto di un immagine, possono percorrere la fotografia lungo tutto la sua dimensione.
  Solitamente non mostrano una direzione deterministica, ma si spostano in modo casuale lungo la stampa.
  Se sono presenti più crepe su una fotografia, ognuna assume la propria direzione.

Fig. 3.2 - Crepe nell'immagine

- Le vecchie fotografie presentano spesso numerosi graffi. Questi sono rappresentati da sottili linee dritte senza alcuna direzione preferenziale.

Fig. 3.3 - Graffi sull'immagine

- Col passare degli anni, le fotografie sono spesso piegate oppure ne vengono girati gli angoli, o ancora vengono accidentalmente accartocciate.
  Queste pieghe possono trasformarsi in vere e proprie fratture e, dipendentemente dall'ampiezza della spaccatura, la varie parti dell'immagine che la circondano possono non coincidere perfettamente.
  Quando le immagini vengono scannerizzate, il difetto appare simile ad una crepa. Inoltre, se alcuni pezzi sono persi, si possono formare dei buchi nell'immagine.

Fig. 3.4 - Fotografia piegata in corrispondenza del volto di una donna

- Infine si possono avere immagini irrimediabilmente modificate dall'uomo con l'aggiunta di materiale estraneo a quello fotografico (spago, graffette, colle, etc.)

Fig. 3.5 - Particolare di fotografia modificata con l'aggiunta di spago

Danni chimici

Tutti i difetti che appartengono a questa categoria sono originati chimicamente.

- Macchie semi-trasparenti, spesso generate da acqua o umidità.
  In questo tipo di difetto, ogni pixel coperto dalla macchia contiene sia informazione sull'immagine originale che rumore.

Fig. 3.6 - Macchia semi-trasparente

- Spaccature sull'immagine generate da attacchi di agenti chimici alla gelatina. Sono indicate come un tipo particolare di macchie, che creano regioni con un omogeneo livello di grigio. Possono divenire molto critiche quando coprono particolari posizioni della fotografia, come i volti.
  In questi casi il recupero dell'immagine originale è poco probabile.

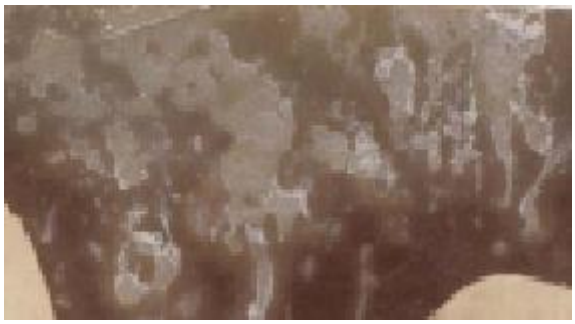


Fig. 3.7 - Attacchi chimici che distruggono l'informazione originale

- Un particolare danno chimico prende il nome di foxing : in alcune fotografie è possibile notare dei punti di colore rossastro-marrone, le cui dimensione possono variare. Questo fenomeno è il risultato di reazioni chimiche tra la stampa fotografica e alcuni microorganismi.

Fig. 3.8 - Immagini affette da " foxing "

Danni da deposito

Con il passare degli anni possono depositarsi sulle fotografie diversi tipi di materiali, siano essi visibili a occhio nudo, sia composti da micro elementi, in modo proporzionale allo stato di conservazione delle immagini stesse.
Questi materiali sono responsabili della creazione di piccoli punti che possono ricoprire l'intera superficie della stampa.

Fig. 3.9 - Punti che coprono l'informazione originale

4. Cenni di restauro manuale

Il restauro della fotografia è una disciplina relativamente recente ed in continuo sviluppo nell'ambito della quale vengono ogni giorno sperimentate nuove tecniche e svolti nuovi studi proprio per riuscire a salvare e conservare il meglio possibile questo manufatto estremamente fragile.
Prima di affrontare un qualsiasi intervento, è chiaramente necessario conoscere la tecnica di manifattura per poter utilizzare i materiali più idonei al suo restauro.
In generale, la prima operazione consiste nella pulitura a secco dell'oggetto in tutte le sue parti per consentire l'eliminazione, o almeno l'attenuazione, di fattori quali polvere, sporco e macchie superficiali che a lungo andare potrebbero creare problemi seri all'immagine. Spesso, infatti, la cattiva conservazione di un'opera in un ambiente causa la sua totale o parziale perdita; ad esempio quando l'ambiente è troppo umido l'acqua contenuta nell'aria favorisce la solubilizzazione dello strato dell'emulsione, o del legante che può in questo modo inglobare al suo interno le particelle che si erano depositate sulla superficie, creando un danno irreparabile, come descritto nel capitolo precedente.
Questa operazione, eseguita a secco con attrezzature idonee e calibrate, è una delle più delicate perché, se eseguita troppo energicamente, può provocare graffi o distacchi dell'emulsione ancora più dannosi della polvere, se al contrario è insufficiente, può alimentare lo sviluppo di microrganismi parassitari.
Nel caso di sostanze più difficili da trattare e rimuovere a secco, come il fango, si possono eseguire delle puliture con solvente specifico, sempre con la necessaria attenzione alla solubilità dello stesso legante nella soluzione impiegata; il lavaggio può essere condotto, ad esempio, con una soluzione a base di acqua demineralizzata e alcool etilico puro in modo da non arrecare danno all'emulsione sottostante.

Nel caso, invece, abbastanza frequente, di distacchi di emulsioni o di mascherature cartacee si può intervenire con infiltrazioni di adesivo a base acquosa o alcolica.
Il legante utilizzato per far aderire lo strato di emulsione ad esempio ad una lastra di vetro negativa tende con l'invecchiamento a distaccarsi, a creare bolle d'aria o frammentarsi soprattutto se è stata incollata sopra di esso una carta sagomata, in modo da creare in stampa una superficie perfettamente bianca ed uniforme (carta barinata).
L'intervento nelle fotografie di vario genere deve essere effettuato su ambo i lati; il *recto*, cioè la parte frontale dove è presente l'immagine, ed il *verso*, cioè la faccia posteriore del supporto.
Per quanto riguarda i danni chimici creati dalla manifattura della fotografia, cioè sbiadimento, ingiallimento e foxing, purtroppo ben poco è oggi possibile fare; l'unica soluzione resta quella di frenare il più possibile il fenomeno con una corretta conservazione dell'opera in un ambiente perfettamente climatizzato e in contenitori idonei dopo aver pulito la superficie con pennelli e gomme.
È chiaro che lo scopo di un intervento di pulitura è quello di eliminare tutte le sostanze che possono interagire col materiale composito della fotografia e di rendere più leggibile l'immagine, ma bisogna anche sapere fino a che punto spingersi senza arrecare inutile stress ad un oggetto già estremamente provato.
I danni causati da fattori meccanici come strappi, tagli o lacune vengono trattati essenzialmente come quelli presenti su materiale cartaceo classico e cioè eseguendo un risarcimento con velina e carta giapponese (o con carta occidentale) e con un adesivo idoneo per la singola tecnica da restaurare. La sola accortezza da adottare, infatti, riguarda l'adesivo che non deve assolutamente creare tensioni o solubilizzare l'emulsione.
Un discorso a parte merita il ritocco, che non si può considerare un intervento conservativo vero e proprio, ma piuttosto estetico, nel senso che ha

essenzialmente la funzione di migliorare la leggibilità di un'opera, attenuando quei fattori che potrebbero distogliere l'attenzione dal soggetto.
Nell'eventualità che, per l'azione di microrganismi, d'insetti, di roditori o dell'uomo, alcune porzioni d'immagine vengano a mancare, si può intervenire con un ricoprimento tramite pittura delle zone lacunose molto vistose, in vari modi.
Il più classico prevede un'integrazione cromatica all'acquerello, tempera o pastello, in modo da non far più risaltare la lacuna. Il problema però è dato dalla superficie della fotografia che spesso è lucida, presenta iridescenze tipiche di tecniche, come ad esempio il collodio, o fattori di degrado come lo specchio d'argento. Non è sempre facile riprodurre l'aspetto originale della superficie, anche con la più grande esperienza.
Si deve prima di tutto ben isolare lo strato sottostante di carta, poiché il colore sciolto in una soluzione acquosa potrebbe penetrare troppo all'interno delle fibre causando macchie. Per fare ciò, si può intervenire con adesivi a base di solventi che non contengono acqua e che hanno l'ulteriore funzione di consolidare gli eventuali distacchi di emulsione lungo i bordi della lacuna da trattare in modo da impedire eventuali infiltrazioni di colore. Su alcuni tipi di fotografie in cui viene utilizzata la così detta *carta barinata* come supporto, cioè una carta ricoperta da uno strato di barite che serve ad isolare lo strato cartaceo dall'emulsione e che, dato il suo particolare colore bianco, riesce a ricreare forti contrasti nelle stampe bianco-nero, si possono ritrovare lacune che è possibile ripristinare con del solfato di barite e poi colorarlo con pennelli od aerografo, strumento questo ultimo indicato per ricreare zone uniformi di colore.
Questi sono solo alcuni esempi d'intervento che è possibile effettuare sul materiale fotografico e che danno quindi un'idea di ciò che oggi si può fare sperando che la tecnologia presto dia ancor più strumenti utili a preservare un oggetto molto importante sia dal punto di vista storiografico che artistico.

5. Cenni di conservazione dei supporti originali

Come si è visto prima una delle cause più lesive è quella della mancanza di cura del materiale che può danneggiare irreparabilmente un'opera considerata in qualsiasi delle sue svariate forme sia essa cartacea o vitrea, o accelerare i processi di degrado fisiologico dell'oggetto fotografico.
Per questo motivo l'IFLA, International Federation of Library Associations [2], ha stabilito delle direttive da seguire per il corretto mantenimento e conservazione non solo delle opere librarie, ma anche della documentazione fotografica su carta e pellicola.
Ecco un breve elenco di alcuni degli accorgimenti descritti in [2]:

- Contenitori composti da materiali contenenti sostanze sulfuree od organiche quali PVC possono intaccare supporti od emulsioni e per questo devono essere utilizzati contenitori di cartoncino ad elevato contenuto di cellulosa (87%) o plastica, ed in entrambe i casi con sostanze a Ph neutro, prive di lignina, riserva alcalina, particelle metalliche, perossidi ed impurità varie;

- L'ambiente di immagazzinamento deve prevedere livelli di umidità adeguati (umidità relativa al 30-40%);

- Le temperature dovranno essere inferiori ai 18 °C per le stampe in bianconero e sotto i 2 °C per quelle a colori;

- Si devono adottare misure per ridurre l'esposizione alla luce, alle radiazioni ultraviolette, all'inquinamento atmosferico e pulviscolare;

- La manipolazione deve avvenire in ambienti appositi privi di agenti inquinanti e dannosi per il materiale fotografico;

- Si dovranno adottare guanti di cotone e si dovrà approntare una superficie pulita per l'appoggio.

Queste sono solo alcune delle norme stilate che fanno capire la fragilità dei supporti fotografici e le accortezze da seguire per poterne prolungare il più possibile la vita. Per avere dettagli maggiori si rimanda comunque al testo completo [2].

6. Identificazione dei difetti

Nel capitolo 3 abbiamo introdotto i principali danni subiti dal materiale fotografico.
Questi difetti, una volta scannerizzata l'immagine, sono molto complessi da identificare in modo automatico, per via della loro tendenza a confondersi con parti dell'immagine stessa che invece non hanno subito alcun danno.
Per spiegare queste caratteristica, prenderemo a esempio un tipo particolare di difetto, le *crepe*.
La crepa è un difetto che lede gravemente il contenuto informativo dell'immagine e, per sua natura, distrae l'osservatore dall'opera in quanto, spesso, essa attraversa la stampa per tutta la sua superficie come si può vedere nella figura 6.1.

Fig. 6.1 - Esempio di stampa fotografica affetta da crepa lungo tutta la sua estensione

Le crepe presentano una grande varietà di caratteristiche che rendono difficile la loro eliminazione o attenuazione tramite restauro sia manuale che digitale.
Solitamente si può notare che questo difetto si sviluppa lungo un'asse principale, dove il supporto è stato ripiegato su se stesso e dove è presente il danno più grande in quanto è la zona che è stata maggiormente sollecitata meccanicamente. Qui si possono trovare linee marcate che testimoniano la piega dell'emulsione, arricciamenti e sormontamenti dell'emulsione su se stessa dovuti

al suo distacco dal supporto e alla pressione esercitata nel momento in cui la fotografia è stata piegata, lacerazioni della parte sensibile con conseguente lacuna informativa ed esposizione alla luce del supporto sottostante che viene sottoposto ulteriormente ad agenti degradanti quali umidità, parassiti, muffe, batteri ed agenti chimici presenti nell'aria creando in questo modo buchi e viraggi del colore originale.
Anche il supporto, quindi, presenta diverse caratteristiche: rugosità, variazioni di colorazione (zone più chiare ed altre più scure, viraggi rossi ed altri gialli a seconda dell'agente inquinante che ha impregnato la sua struttura), presenza di particelle che ne limitano l'uniformità e così via.
È quindi difficile discernere con precisione l'azione da prendere, non solo su crepe diverse, ma addirittura sulla stessa crepa perché a distanze relativamente piccole possono verificarsi condizioni del tutto differenti.
Spesso nella periferia di questa zona critica s'instaurano ulteriori difetti conseguenze dei primi: micro-lesioni, micro-crepe, micro-lacerazioni dovuti alla spinta esercitata dall'emulsione maggiormente sollecitata sulle parti confinanti o di contro, quando l'emulsione è stata tenuta per periodi lunghi di tempo in posizione piegata ed essendosi adattata a tale posizione, una volta riportata alla situazione originaria e corretta, strappa lembi di materiale sensibile al proprio contorno; inoltre, si possono creare delle lacerazioni del materiale sensibile per il fatto che, una volta piegata la fotografia, le parti di emulsione sovrapposte si attaccano tra loro e quando la fotografia viene riportata nella corretta posizione le parti già lese dalla prima azione meccanica e quindi già staccate dal supporto rimangono attaccate tra loro creando lacune da una parte e sormontamenti di materiale sensibile dall'altra.
La seguente figura riassume tutte le tipologie di difetti riscontrabili appena descritti.

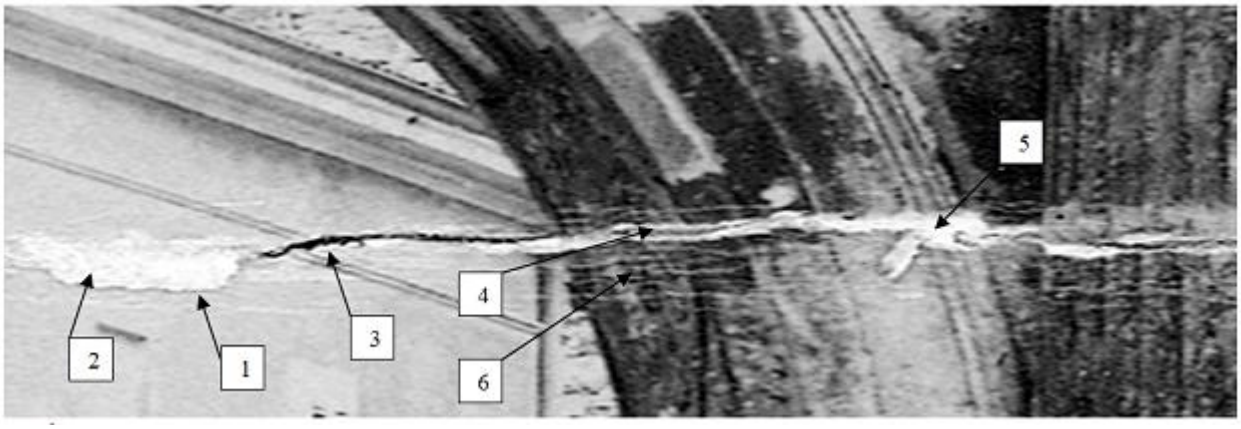


Fig. 6.2 - Particolare dell'immagine precedente, in cui sono riscontrabili alcune caratteristiche tipiche delle crepe

In dettaglio:

1. Pieghe ad arricciamenti dovuti al distacco dell'emulsione dal supporto ed il conseguente strappo ed arricciamento;

2. Zona chiara uniforme data dal supporto barinato che è venuto alla luce per la mancanza di emulsione. Questa colorazione testimonia una recente esposizione all'aria in quanto gli agenti inquinanti presenti nell'aria non sono ancora riusciti ad intaccare la proprietà di tale sostanza ovvero il fatto di avere un colore bianco molto intenso ed uniforme;

3. Lacerazione totale dell'emulsione e del supporto. Qui non solo viene a mancare il contenuto informativo, come nel caso precedente, ma si è formata anche una lacuna del supporto cartaceo creando un solco nero dato dalle ombre proiettate dai lembi di carta strappati ed inoltre si vede che i lati dell'immagine opposti alla crepa non coincidono tra loro;

4. Variazione di tonalità. In questa regione si può notare come il supporto acquisti differenti tonalità: più scure in prossimità dell'asse principale, dove il supporto è stato maggiormente sollecitato e danneggiato e quindi più facilmente attaccabile da agenti chimici presenti nell'aria; più chiare dove la sollecitazione è stata inferiore;

5. Sbiadimento. L'emulsione, una volta distaccata dal supporto ed essendo stata maggiormente esposta all'aria, ha perso intensità risultando sbiadita e virando la propria colorazione in un grigio visibilmente differente dall'emulsione confinante ancora incollata al supporto che le fornisce protezione;

6. Micro-fratture periferiche. Non fanno direttamente parte del danno principale, in quanto ben lontane da esso, ma la loro struttura longitudinale richiama la deformazione indotta meccanicamente. Queste sono state indotte dalla pressione esercitata dall'emulsione principalmente sollecitata sulle aree limitrofe che si sono, di conseguenza, distaccate dal supporto ed arricciate.

Proprio per via di queste moltitudine di difetti introdotti dalle crepe, l'identificazione automatica è una strada che, per ora, è convenientemente trascurabile.
L'alternativa consiste nel richiedere un minimo sforzo da parte dell'utente, consegnando nelle sue mani la responsabilità di segnalare la zona affetta dal danno, evidenziandola in maniera molto semplice, per esempio tramite un pennello.
Si consideri un'immagine come la seguente

Fig. 6.3

in cui il difetto si confonde, quasi a mimetizzarsi, con le divise dei tra personaggi.
Un utente non avrebbe alcun problema a selezionare la parte di immagine da dedicare al restauro.
Un algoritmo di identificazione automatica, per esempio [3], che descrive un metodo per poter rintracciare le crepe nei vecchi dipinti e affreschi basato principalmente su un filtro morfologico operante in scala di grigi denominato "top-hat transform"[4] e su una classificazione ottenuta attraverso l'uso di una rete neurale basata su funzione radiale mediana (MRBF) [5], fallirebbe, invece, nella ricerca della crepa presente nell'immagine.
Non tratteremo la teoria relativa all'algoritmo accennato, in quanto non rientra nei nostri scopi, mostriamo solamente un semplice esempio di fallimento da parte di metodo nella ricerca di una crepa.
Questo è visibile nella seguente immagine:

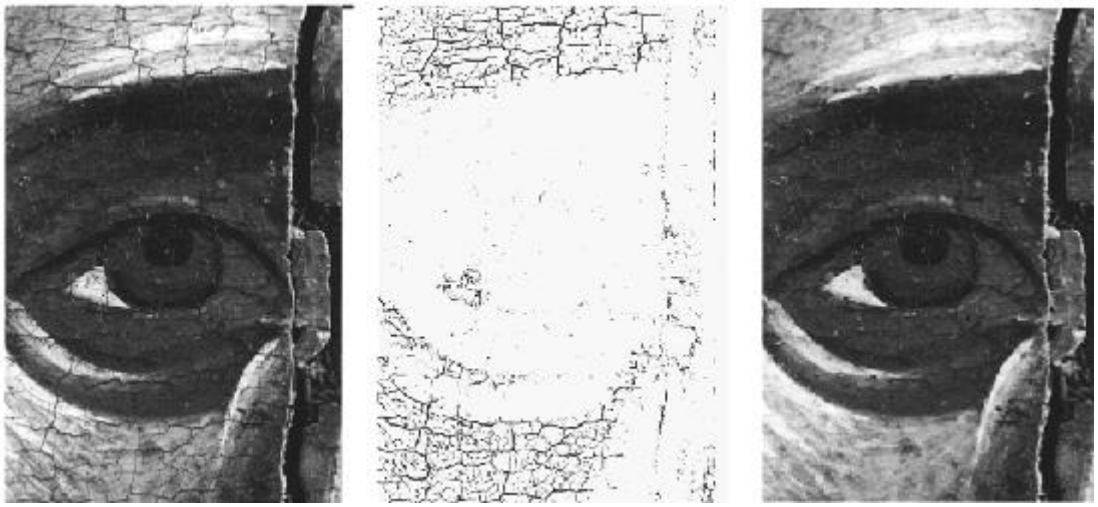


Fig. 6.4

Le crepe attorno agli occhi, ovvero in zone in intesità luminosa simile a quella del difetto, non sono state completamente identificate dall'algoritmo.

7. Restauro digitale

Come già accennato, la fotografia è una forma d'arte relativamente recente, ma è probabilmente la più diffusa nel mondo. La sua introduzione nella vita quotidiana ci ha permesso di creare un'ampia quantità di documentazione di elevato valore culturale.
Sfortunatamente le vecchie stampe fotografiche erano (e sono) basate su fragili materiali, influenzabili dalle cattive condizioni ambientali. Per questa ragione, negli ultimi anni, l'attenzione alla conservazione e alla preservazione delle stampe è notevolmente aumentata.
Tuttavia rimane ancora moltissimo materiale fotografico danneggiato che necessita di restauro. Le classiche tecniche di restauro fisico, manuale, sono estremamente costose e spesso il danno non può essere del tutto rimosso.
La diffusione di scanner e software per la manipolazione di immagini, ha aperto un nuovo mondo di recupero del materiale fotografico deteriorato. Le vecchie immagini, dopo essere state digitalizzate, possono essere restaurate virtualmente e, se necessario, ristampate, originando una copia restaurata dell'immagine iniziale. L'obiettivo è quello di rendere irriconoscibile, all'occhio di un utente ignaro del restauro, quale sia l'immagine originale, senza danno, e quale quella restaurata, rendere cioè "invisibile" l'intervento dell'elaboratore.
È chiaro che prima di procedere ad un restauro digitale è necessario intervenire, almeno in un primo momento, in modo manuale per eliminare elementi di disturbo come la polvere, fango e altre sostanze superficiali che oltre a non far parte del supporto vero e proprio potrebbero creare, inutilmente, dei problemi di ricostruzione dell'immagine.
Dopodiche si interviene in modo virtuale, sui difetti che sono ormai diventati parte del supporto stesso e che non potrebbero essere rimossi altrimenti come ad esempio macchie che coinvolgono sia lo strato sensibile sia il supporto, oppure impronte digitali, crepe, pieghe, lacerazioni e così via.

Esistono alcuni software commerciali che propongono tecniche di restauro guidato dall'utente, dove i difetti non sono automaticamente identificati e tutte le correzioni devono essere suggerite dall'operatore. In questo modo il restauro diviene complesso e costoso, poiché solamente personale esperto è in grado di portare a termine l'operazione di correzione del difetto.
Sono quindi richieste tecniche automatiche per ottenere un restauro semplice, veloce ed efficiente. Il resaturo digitale può essere vista come un'operazione di filtraggio, in cui l'immagine originale fornita in ingresso viene modificata dall'algoritmo e portata in uscita.

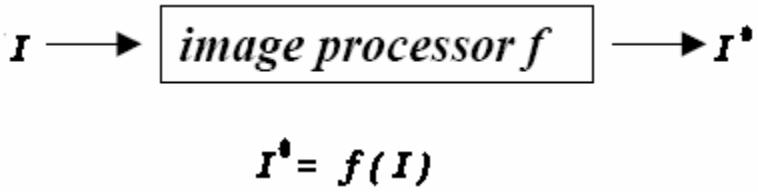


Fig. 7.1 - Tipico processo di filtraggio

Nei capitoli successivi verrà trattata in dettaglio la tecnica di restauro digitale e verranno riportati alcuni metodi particolari utilizzati per porre rimedio al problema della mancanza di informazione nelle immagini e che più di altri si adattano allo scopo che ci siamo prefissi, l'eliminazione dei difetti dalle stampe fotografiche; come si vedrà i metodi presentati traggono origine da diverse teorie elaborate inizialmente per scopi di tutt'altro genere, ma che si sono rivelate utili anche allo scopo dell'elaborazione digitale ed in particolare al restauro delle immagini.
Verranno trattate le relative teorie, mettendo in luce gli aspetti innovativi e utili allo scopo del restauro e evidenziando i difetti riscontrati.

8. Maschere

A causa delle difficoltà nel rilevamento automatico dei difetti, analizzate nel capitolo 6, la scelta più conveniente in quanto semplice e veloce, per identificare l'area affetta da danno, consiste nel consegnare la sua selezione nelle mani dell'utente.
Questo procedimento può essere portato a termine con uno qualsiasi dei più comuni software di disegno; basta infatti aprire l'immagine desiderata e colorare la parte che si vuole sottoporre all'algoritmo.
Si ottiene così una nuova immagine, chiamata maschera.
Questa dovrà essere nera lungo il difetto e bianca per tutto il resto dell'immagine.
Durante l'esecuzione dell'algoritmo, questi andrà a cercare sull'immagine maschera i punti di colore nero e opererà solamente su quelli, trascurando tutto il resto.

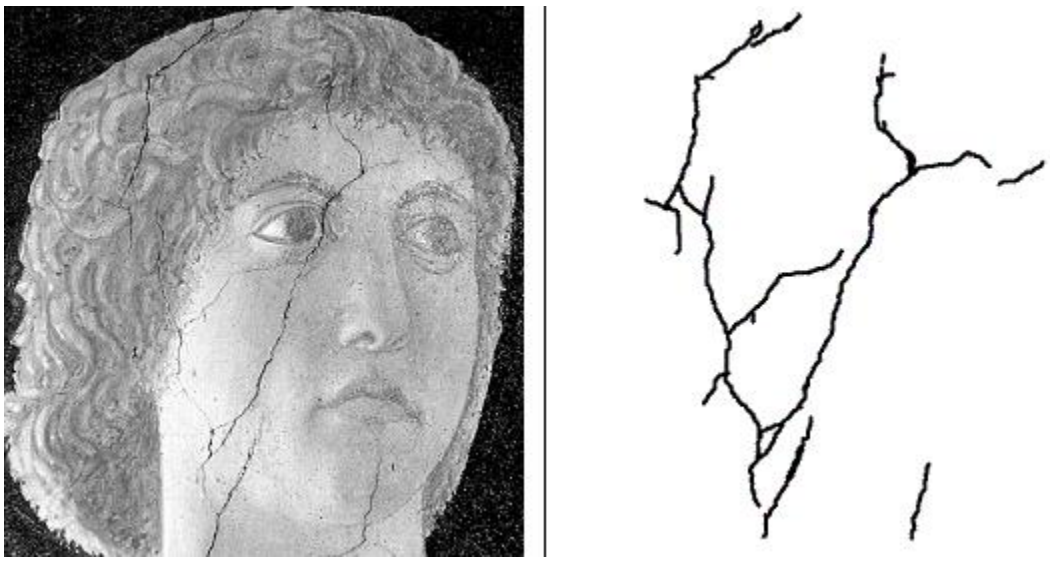


Fig. 8.1 - Immagine affetta da crepe e relativa maschera

9. Inpainting

Finora abbiamo parlato di restauro di materiale fotografico d'epoca affetto da danni causati dall'usura dei supporti, dovuta al tempo e alla cattiva conservazione. Ma le tecniche di restauro che presenteremo non hanno come unico compito quello di "risanare" vecchie fotografie, esistono altri campi di utilizzo, quale il foto ritocco o gli effetti speciali, in modo particolare su pellicole cinematografiche, dove ogni frame è considerato come un'immagine indipendente dalle altre. Vediamo alcuni esempi.

Fig. 9.1 - Immagine a colori con testo

In casi come questo, in cui un testo "super-imposto", ricopre gran parte dell'immagine, una eventuale richiesta di una sua eliminazione, in modo da ottenere un'immagine pulita e senza rumore aggiunto, è un compito delicato. Questo perché il testo, oltre a ricoprire zone omogenee come la strada, è posizionato su aree molto dettagliate dell'immagine, come il carretto con il cocchiere e il cavallo.

Il risultato a cui si vuole arrivare è il seguente:

Fig. 9.2

Quest'ultima immagine è la copia restaurata della precedente, ma se sottoposta ad un'utente ignaro, potrebbe benissimo sembrare l'immagine originale, prima che le si applicasse il testo.
Un'altra semplice applicazione degli algoritmi di restauro al campo degli effetti speciali risiede nella rimozione di oggetti o personaggi indesiderati. Si pensi alla registrazione di una scena di un film, in cui è accidentalemente inserito un operatore o quando cavi e microfoni rimangono all'interno dell'area di ripresa della videocamera. Solitamente ci si accorge di questi errori durante il montaggio. A quel punto le possibilità sono: ripetere nuovamente la scena, processo troppo costoso, oppure ignorare completamente la svista, sperando che nessuno si accorga di nulla, scelta, anche questa, non certamente preferibile. Un'alternativa è quella di passare i frame in cui compare l'oggetto indesiderato ad un semplice software di inpainting, questo eliminerà l'intruso senza lasciare traccia, in modo da trasformare l'immagine:

Fig. 9.3 - Scena di film in cui è rimasto incluso un microfono (cerchiato di rosso)

nella seguente:

Fig. 9.4

Si nota come nell'immagine modificata non è presente alcun segno dell'eliminazione del microfono.

Un algoritmo che esegue operazioni come quelle elencate è chiamato algoritmo di *inpainting*. Il termine "inpainting" è stato coniato nel 2000 da M. Bertalmio e G. Sapiro [7], pionieri di questa tecnica, ed è l'acronimo di interpolated painting, ovvero dipingere attraverso l'uso dell'interpolazione.
Esaminiamo a fondo l'algoritmo proposto in [7].

Image Inpainting

La modifica di immagini in modo che un utente che non sia a conoscenza dell'immagine originale non possa notare differenze tra questa e quella restaurata, è una tecnica molto antica, opere medievali iniziarono a essere restaurate già prima del rinascimento, in modo da riempire eventuali mancanze di colore negli affreschi e nei dipinti. Questa tecnica è chiamata ritocco o inpainting.
La necessità di ritoccare le immagini si è poi estesa, con la loro invenzione e introduzione, alla fotografia e alla cinematografia. Ma l'obiettivo rimane lo stesso, invertire il processo di deterioramento dei supporti (per esempio le crepe nelle fotografie o i graffi e punti nelle pellicole) oppure di rimuovere elementi di disturbo.
L'algoritmo di image inpainting si basa proprio sulle idee e sulle tecniche utilizzate dai restauratori manuali professionisti.
Prima di tutto notiamo che i classici algoritmi per l'eliminazione del rumore nelle immagini non sono applicabili a questo tipo di problematiche, poiché nella comune elaborazione di immagini, ogni pixel "difettoso" contiene sia parte del dato corretta, sia rumore aggiuntivo. Questo non accade nelle immagini a cui si richiede di applicare l'inpainting, qui infatti i pixel del difetto non danno alcuna

informazione riguardo alla loro vera natura, l'unica informazione disponibile è quella che circonda il danno.
Consideriamo di indicare con Ω la regione affetta dal danno e che deve essere sottoposta all'algoritmo di inpainting, notando che nessuna assunzione è fatta sulla tipologia di questa regione, indichiamo con $\partial\Omega$ il contorno di questa regione.
Intuitivamente, la tecnica proposta da Bertalmio et al. tende a prolungare le linee delle figure lungo la direzione ortogonale al gradiente (isophote) che incontrano il contorno della regione danneggiata, come possiamo vedere nell'immagine:

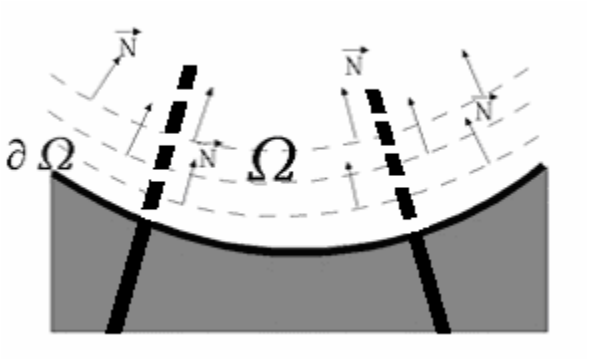


Fig. 9.5

la linea tratteggiata rappresenta il contorno di Ω che man mano che l'algoritmo procede, si sposta sempre più verso l'interno. Questo proprio perché si cerca di portare l'informazione che circonda il difetto dall'esterno di esso, verso l'interno, cercando di mantenere l'angolo di incidenza che le isophote hanno quando incontrano $\partial\Omega$.
Prima di presentare in dettaglio l'algoritmo, cerchiamo di capire come operano gli artigiani professionisti che si occupano di restauro. Gli esperti in conservazione dell'Istituto d'Arte di Minneapolis, consultati da Bertalmio et al. per l'occasione,

chiarirono innanzitutto che l'inpainting è una tecnica molto soggettiva, dipendente da restauratore a restauratore, ma stillarono una metodologia generale suddivisa in quattro punti:

1. L'immagine globale determina il modo in cui è necessario riempire la parte coperta da difetto, lo scopo dell'inpainting diventa così ri-ottenere un'immagine completa che mantenga la sua unità se osservata tutta intera;

2. La struttura dell'area che circonda Ω è continuata all'interno della zona in esame, le linee di contorno sono disegnate cercando di prolungare quelle che incontrano $\partial\Omega$;

3. Le differenti regioni all'interno di Ω sono riempite con colori in accordo con quelli dell'intorno $\partial\Omega$;

4. I piccoli dettagli sono disegnati manualmente una volta teminati i tre punti precedenti.

Immediatamente si può imparare molto da queste regole di base.
L'algoritmo proposto in [7] esegue iterativamente e simultaneamente i punti (2.) e (3.). Cerchiamo ora di tradurre i concetti legati all'inpainting manuale in modo da utlizzarli nel nostro contesto, esprimendo il tutto tramite rigorose formule matematiche. Innanzitutto consideriamo la funzione $I_0(i,j):[0,M]\times[0,N]\to\mathbb{R}$ con $[0,M]\times[0,N]\subset\mathbb{N}\times\mathbb{N}$ dove I è una funzione bidimensionale che, data un'immagine di dimensione M *x* N restituisce il valore del pixel alla posizione *(i, j)* .
Dalla descrizione della tecnica di inpainting manuale, la scelta naturale per questo algoritmo sembra essere una soluzione di tipo iterativo.
La procedura di inpainting digitale costruirà una famiglia di immagini

$$I(i,j,n):[0,M]\times[0,N]\times\mathbb{N}\to\mathbb{R}$$

dove ognuna delle *n* immagini è data dalla riduzione di un livello, ottenuto portando l'informazione esterna a Ω nel suo interno.

Fig. 9.6 - Notare l'evoluzione dell'algoritmo, la prima immagina a sinistra è quella difettosa, man mano che ci si sposta verso destra si tende al risultato finale, dato nell'ultima immagine

La famiglia di immagini generate è tale che $I(i,j,0)=I_0(i,j)$ e $\lim_{n\to\infty} I(i,j,n)=I_R(i,j)$ ovvero, la prima immagine della famiglia è proprio l'immagine iniziale, mentre l'*n*-esima corrisponde con il risultato finale dell'algoritmo, $I_R(i,j)$.
Ogni algoritmo di inpainting può essere scritto nella forma

$$I^{n+1}(i,j)=I^n(i,j)+\Delta t I_t^n(i,j),\forall(i,j)\in\Omega \qquad (1)$$

dove il super-indice *n* denota il "tempo", *(i,j)* sono le coordinate del pixel, *Δt* è il tasso di miglioramento e $I_t^n(i,j)$ indica l'aggiornamento dell'immagine $I^n(i,j)$.
Con questa equazione $I^{n+1}(i,j)$ risulta essere una versione migliorata di $I^n(i,j)$, al crescere di *n* otteniamo immagini sempre migliori. Il nostro obiettivo è quindi descrivere $I_t^n(i,j)$, ovvero il miglioramento dell'immagine ad ogni passo dell'algoritmo.

Consideriamo $L^n(i,j)$ come l'informazione esterna al difetto e che vogliamo propagare e $\vec{N}^n(i,j)$ la direzione di propagazione, questo significa che dobbiamo avere:

$$I_t^n(i,j) = \delta\vec{L}^n(i,j) \cdot \vec{N}^n(i,j) \qquad (2)$$

dove $\delta\vec{L}^n(i,j)$ rappresenta la quantità di cambiamento nell'informazione $L^n(i,j)$ proveniente dal passo precedente.

Con questa equazione stimiamo l'informazione $L^n(i,j)$ della nostra immagine e computiamo il suo cambiamento lungo la direzione $\vec{N}$.

Notiamo che quando l'algoritmo converge si ha $I^{n+1}(i,j) = I^n(i,j)$ e, dalla (1) e dalla (2), abbiamo $\delta\vec{L}^n(i,j) \cdot \vec{N}^n(i,j) = 0$, che significa esattamente che l'informazione L è stata propagata lungo la direzione di $\vec{N}$.

Dato che vogliamo ottenere la propagazione tramite diffusione, L può essere un qualsiasi stimatore di sfocamento.

Noi utilizzeremo una semplice implementazione del laplaciano, definita come $L^n = I_{xx}^n(i,j) + I_{yy}^n(i,j)$.

Vediamo come può essere approssimato il Laplaciano.

Possiamo scrivere la derivata prima parziale lungo la direzione di *x* come

$$\frac{\partial I(x,y)}{\partial x} \approx I(x+1,y) - I(x,y)$$

e

$$\frac{\partial I(x,y)}{\partial x} \approx I(x,y) - I(x-1,y)$$

Dalla differenza delle due otteniamo la versione discreta della derivata seconda parziale di $I(x,y)$ nella direzione di *x*:

$$\frac{\partial^2 I(x,y)}{\partial^2 x} \approx 2I(x,y) - I(x-1,y) - I(x+1,y)$$

Lo stesso approccio può essere usato per ricavare la prima e la seconda derivata parziale di $I(x,y)$ nella direzione di *y*:

$$\frac{\partial I(x,y)}{\partial y} \approx I(x,y+1) - I(x,y) \quad \text{e} \quad \frac{\partial I(x,y)}{\partial y} \approx I(x,y) - I(x,y-1)$$

da cui

$$\frac{\partial^2 I(x,y)}{\partial^2 x} \approx 2I(x,y) - I(x,y-1) - I(x,y+1)$$

Combinando insieme le derivate seconde otteniamo il nostro operatore Laplaciano:

$$L^n(x,y) = 4I(x,y) - I(x-1,y) - I(x+1,y) - I(x,y-1) - I(x,y+1)$$

Se L^n indica la di variazione di colore in un punto, il vettore $\vec{\delta} L^n$ indica la quantità di cambiamento, di variazione, per quel determinato pixel.
Un altro importante operatore citato è il gradiente, che indica la direzione di massimo incremento di una funzione in *n* variabili. Nel nostro caso la funzione è in due variabili (*x* e *y*) e rappresenta il colore dell'immagine.
Mentre il Laplaciano è definito sulle derivate seconde, la definizione di gradiente si basa sulle derivate prime:

$$\nabla I(x,y) = \begin{bmatrix} I_x(x,y) \\ I_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

Notiamo che è ottenuto dalla derivata della distribuzione dei livelli di grigio in entrambe le direzioni, *x* e *y*, formando un vettore bidimensionale che può essere identificato dal suo modulo e dalla sua direzione. Vedremo successivamente come sono definite queste due componenti. In figura 9.7 è mostrata la sensibilità della derivata prima e seconda alle variazioni di colore

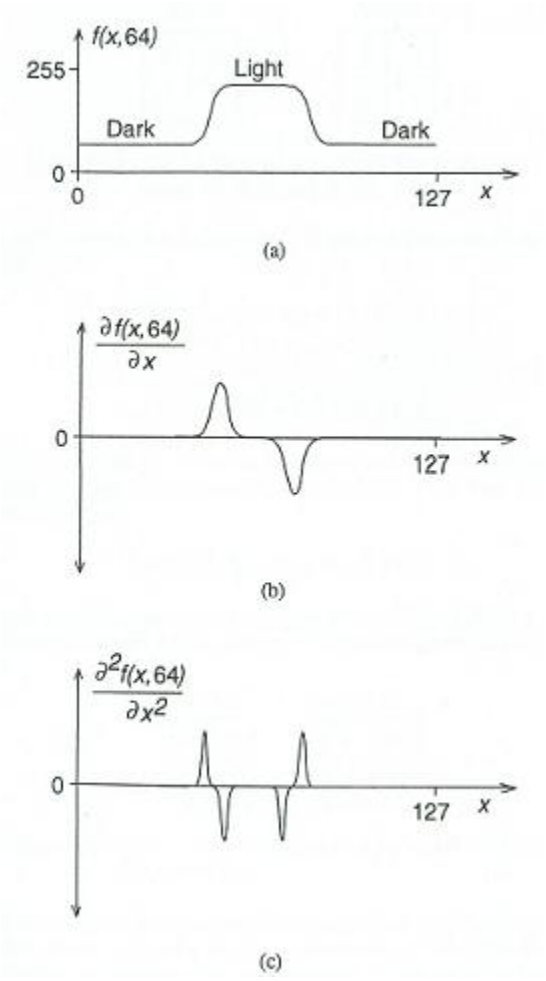


Fig. 9.7 – Sensibilità delle derivate ai cambiamenti di colore

Nella figura 9.7a si mostra un grafico monodimensionale della distribuzione dei livelli di grigio in un'immagine di dimensioni 128x128, alla coordinata *y*=64. Questo rappresenta una regione centrale bianca circondata da regioni scure, ottenendo quindi due bordi di confine.
La figura 9.7b mostra la derivata prima nella direzione di *x*. L'altezza del valore della derivata è proporzionale alla differenza tra le due aree nella zona di confine, per questo può essere utilizzata per identificare la presenza delle isophote.
Nella parte 'c' della figura abbiamo la derivata seconda, questa produce un doppio impulso per ogni "confine", risulta quindi essere più sensibile alle variazioni rispetto alla derivata prima, ma purtoppo è anche più sensibile al rumore.
Le restanti equazioni su cui si basa questo algoritmo sono:

$$\frac{\vec{N}(i,j,n)}{|\vec{N}(i,j,n)|} = \frac{(-I_y^n(i,j), I_x^n(i,j))}{\sqrt{(-I_y^n(i,j))^2 + I_x^n((i,j))^2}}$$

$$\delta\vec{L}^n(i,j) = (L^n(i+1,j) - L^n(i-1,j), L^n(i,j+1) - L^n(i,j-1))$$

$$\beta^n(i,j) = \delta L(i,j)\frac{\vec{N}(i,j,n)}{|\vec{N}(i,j,n)|}$$

$$|\nabla I^n(i,j)| = \sqrt{I_{xbm}^n(i,j)^2 + I_{xfM}^n(i,j)^2 + I_{ybm}^n(i,j)^2 + I_{yfM}^n(i,j)^2} \quad \text{se} \quad \beta^n > 0$$

$$|\nabla I^n(i,j)| = \sqrt{I_{xbM}^n(i,j)^2 + I_{xfm}^n(i,j)^2 + I_{ybM}^n(i,j)^2 + I_{yfm}^n(i,j)^2} \quad \text{se} \quad \beta^n < 0$$

$$I_t^n(i,j) = \beta^n(i,j)|\nabla I^n(i,j)|$$

quindi: prima computiamo lo stimatore lapliaciano L^n e la direzione delle isophote $\vec{N}/|\vec{N}|$ (ortogonale alla direzione del gradiente), dopodichè calcoliamo β^n, la proiezione di $\delta\vec{L}$ lungo il vettore normalizzato $\vec{N}$. Così facendo otteniamo il cambiamento di L lungo la direzione $\vec{N}$. Infine otteniamo il nostro

aggiornamento $I_t^n(i,j)$, moltiplicando β^n per una versione limitata del modulo del gradiente dell'immagine, ∇I. Gli indici *b* e *f* indicano rispettivamente backward e forward, mentre *m* e *M* denotano il minimo e il massimo tra la derivata e zero.

Ricapitolando, stimiamo la variazione dello *smoothing*, data da una discretizzazione del laplaciano bidimensionale e proiettiamo questa variazione lungo la direzione delle isophote. Questa proiezione è usata per aggiornare il valore dell'immagine all'interno della regione affetta dal danno.

Per assicurare una corretta evoluzione dell'algoritmo, viene inserito all'interno dello stesso, un processo di diffusione, ovvero, dopo un certo numero di passi di esecuzione dell'algoritmo di "aggiornamento" dell'informazione, si applicano alcune iterazioni di diffusione, in modo da diffondere le modifiche fino a quel punto apportate all'immagine.

Per effettuare la diffusione si utilizza un procedimento basato sul lavoro di P.Perona e J.Malik, [8], dove si introduce la diffusione anisotropica, processo che permette di diffondere i colori all'interno di una zona uniforme, mantenendo nitidi i contorni.

Nel nostro contesto questo permette di raggiungere l'obiettivo senza perdità di nitidezza nella ricostruzione.

In particolare, le equazioni che sono alla base della diffusione anisotropica sono le seguenti

$$I^{n+1} = I^n + \lambda(c_N \nabla_N + c_S \nabla_S + c_E \nabla_E + c_W \nabla_W)$$

$$\nabla_N = I^n(x, y-1) - I^n(x,y) \qquad \nabla_S = I^n(x, y+1) - I^n(x,y)$$

$$\nabla_E = I^n(x+1, y) - I^n(x,y) \qquad \nabla_W = I^n(x-1, y) - I^n(x,y)$$

$$c_N = g(|\nabla_N|) \qquad c_S = g(|\nabla_S|)$$

$$c_E = g(|\nabla_E|) \qquad c_W = g(|\nabla_W|)$$

Dove gli indici N, S, E e W indicano rispettivamente North, South, East e West e con $0 \le \lambda \le 1/4$.

Sempre in [8] vengono proposte due alternative per la funzione g:

$$g(\nabla I) = e^{(-(\|\nabla I\|/K)^2)}$$

e

$$g(\nabla I) = \frac{1}{1+\left(\frac{\|\nabla I\|}{K}\right)^2}$$

dove più K è grande e più si ha sfuocamento dell'immagine e quindi smussamento dell'aggiornamento e perdita di contorni.

L'algoritmo appena descritto funziona molto bene nella ripristinazione di piccole zone danneggiate, ma, a detta degli stessi autori, causa degli sfocamenti poco gradevoli e innaturali quando la zona sottoposta, coperta dalla maschera, aumenta di dimensioni.

Inoltre non è certamente tra i migliori in quanto a velocità di esecuzione, la complessità temporale è nell'ordine dei minuti.

Vedremo ora un algoritmo studiato con l'intento di migliorare la velocità del precedente.

Fast Digital Image Inpainting (FDII)

L'algoritmo di Bertalmio et al. usualmente richiede parecchi minuti se eseguito sui comuni personal computer, anche per inpainting su aree relativamente piccole. Questi tempi sono talvolta inaccettabili e sono alla base delle motivazioni

che spinsero M.M.Oliveira et al. a sviluppare un algoritmo [9] più semplice e veloce, in grado di produrre risultati molto simili e in pochi secondi.
Una delle assunzioni su cui si basa questo algoritmo risiede nel sistema visivo umano, questo infatti è in grado di "tollerare" una certa quantità di sfuocamento, di smussamento dell'immagine se localizzato in aree non associate ad alto contrasto. Un'altra assunzione invece si basa sulla limitatezza dell'algoritmo [7], che opera bene quando la zone da sottoporre a inpainting è puttosto piccola. Gli autori di FDII infatti affermano che le stesse oparazioni dell'algoritmo "Image Inpainting" su aree limitate, possono essere eseguite utilizzando metodi matematici molto più semplici. Consideriamo ancora Ω, la nostra area da recuparare, e $\partial\Omega$, il suo contorno.
Dato che Ω è piccola, la procedura di inpainting può essere approssimata ad un processo di diffusione isotropica, che propaga l'informazione da $\partial\Omega$ in Ω.
La diffusione isotropica, a differenza di quella anisotropica, sfuoca tutta quanta l'immagine, senza tener conto della conservazione dei contorni.
La versione più semplice dell'algoritmo proposto da Oliveira et al., inizia "pulendo" la zona coperta dalla maschera, ovvero annullando il valore di quei pixel, in quanto inutile al fine della diffusione isotropa. Il passo successivo è quello di eseguire più volte la convoluzione tra i pixel di Ω e un kernel di diffusione.
Questo passaggio di convoluzione con un kernel Gaussiano, per esempio un kernel che effettua la media pesata dei pixel che circondano il pixel in esame, è equivalente alla diffusione isotropica.
Possiamo così riassumere il ciclo iterativo:

```
initialize Ω;
for (iter =0; iter < num_iteration; iter++)
  convolve masked regions with kernel;
```

dove questo ciclo viene ripetuto una prima volta utilizzando il kernel

a	b	a
b	0	b
a	b	a

Primo kernel Gaussiano, a = 0.073235 e b = 0.176765

e poi una seconda volta con

c	c	c
c	0	c
c	c	c

Secondo kernel Gaussiano, c = 0.125

Questa tecnica, per via della sua natura "isotropa", crea degli sfuocamenti nelle zone ad alto contrasto. Nell'articolo stesso [9] è proposta una metodologia che, senza diminuire la velocità dell'algoritmo, è in grado di ovviare al problema.
Nelle zone in cui si desidera mantenere il contorno e non avere l'effetto diffusione, viene inserita una barriera. La barriera non è altro che un piccolo segmento, lungo quanto lo spessore della maschera.
Vediamo nella seguente immagine. La maschera è colorata di giallo.
A sinistra, senza barriere, possiamo notare cerchiati rossi gli sfuocamenti prodotti. A destra, dove le barriere sono di colore rosso, l'immagine è restaurata senza l'effetto di diffusione non desiderato.

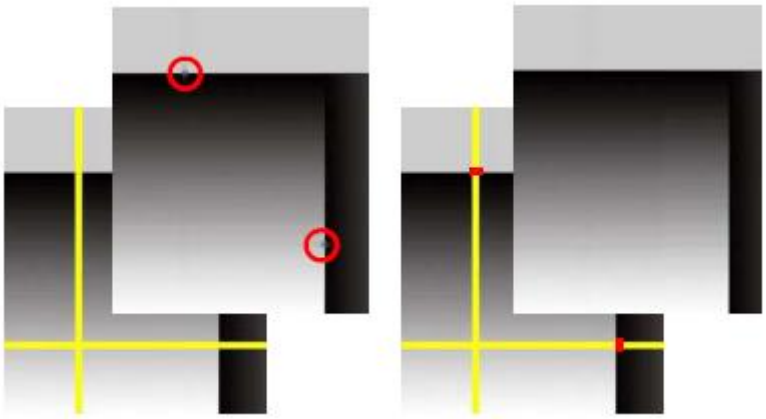
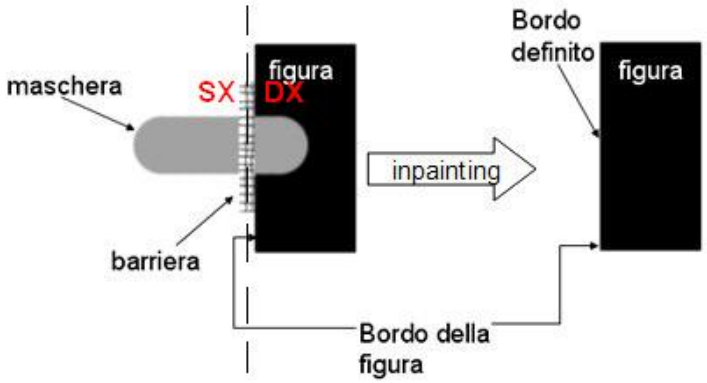


Fig. 9.7 - La maschera è indicata in giallo, a sinistra abbiamo gli sfuocamenti, cerchiati di rosso. A destra vengono utilizzate le barriere (rosse) e il risultato è nettamente migliore

Ma non abbiamo parlato di come le barriere funzionino.
Basiamoci sulla figura:



Se il pixel indagato è sito a sinistra della barriera (la zona a sinistra della linea tratteggiata, indicata con SX), quando l'algoritmo effettua la convoluzione con uno dei due kernel, terrà conto solo dei pixel che circondano il punto in esame e che sono ancora alla sinistra della barriera, non utilizzerà quelli alla destra per il calcolo della media pesata.
Stessa cosa, ma ovviamente invertita, accadrà se ci si trova alla destra della maschera.

In questo modo il pericolo di sfocamento dei bordi è scongiurato.

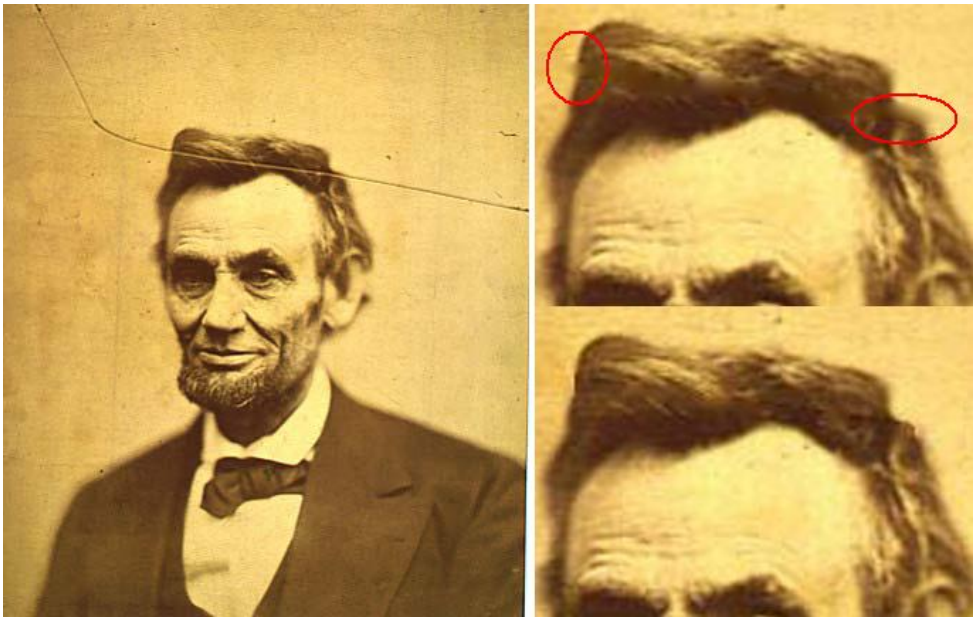


Fig. 9.9 - A sinistra immagine originale con difetto. A destra in alto particolare delli immagine restaurata senza l'uso delle barriere, sono cerchiati gli sfocamenti. Destra in basso restauro con barriere.

L'algoritmo risulta essere nettamente più veloce di quello proposto in [7], ma può risultare complesso per il restauratore il compito di piazzare le barriere in immagini ricche di zone ad alto contrasto, rischiando di ottenere pessimi risultati.
Inoltre, come per "Image Inpainting", quando l'area di restauro aumenta di dimensioni, l'algoritmo non opera in maniera per nulla soddisfacente.

10. Texture Synthesis

Nel campo tessile una “texture” indica un trattamento compositivo grafico nel quale i segni visivi sono disposti in modo tale da formare una superficie equilibrata nell’alternanza di pieni e vuoti, disegno e fondo, ma che non crea alcuna profondità.
In ambito informatico una texture è un’immagine utilizzata per rivestire la superficie di un oggetto virtuale, tridimensionale o bidimensionale, con un apposito programma di grafica.
La texture synthesis è un campo di ricerca ben più vecchio dell’inpainting, da anni utilizzato per particolari applicazioni grafiche.
Le sue radici risalgono agli anni ’80.
Consideriamo le due texture in figura 10.1

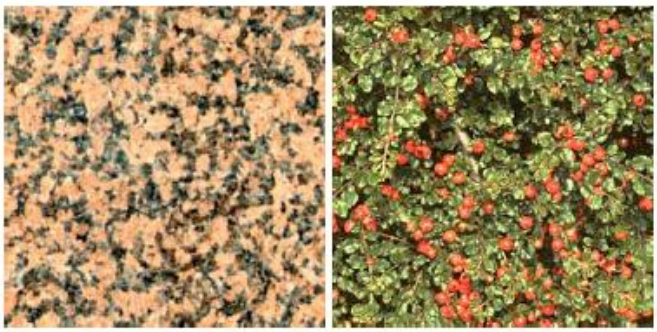


Fig. 10.1 - Esempi di texture

e pensiamo di applicarle al disegno tridimensionale di una teiera, il risultato ottenuto è il seguente

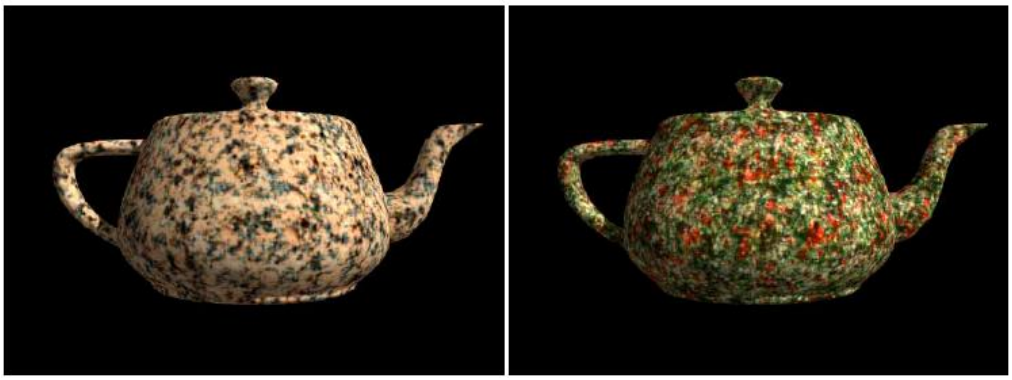


Fig. 10.2

Questo è uno dei principale utilizzi delle texture applicato alla “computer graphic”. Ma non è certamente l’unico. Cerchiamo di capire meglio come funziona la sintesi delle texture e in che modo può essere utilizzata in problemi di restauro digitale. Pensiamo di avere un’immagine che rappresenti uno schema ben preciso come può essere una spiaggia di sabbia, una distesa di sassolini, una prato fiorito, un muro, oppure anche disegni sintetici, realizzati artificialmente, ma che rappresentino degli schemi, ripetuti in modo casuale o ben definiti.

Fig. 10.3 - Altri esempi di texture

e immaginiamo di voler replicare la nostra texture in modo da ottenere, su una nuova immagine, lo stesso andamento, la stessa tessitura, magari applicata ad uno spazio molto più ampio, o solo ad un angolo o che segua uno schema ben preciso.
Per esempio, prendendo i fiori nell'immagine precedente possiamo voler creare, in una nuova immagine, un prato fiorito, come se "attaccassimo" più volte insieme quella stessa immagine, ma in modo da mantenere una struttura corretta, senza l'effetto di cambiamento netto che avremmo copiando e incollando ripetutamente la stessa texture di partenza.
Questo è realizzabile tramite ben definiti algoritmi di texture synthesis.
Tali algoritmi necessitano di ricevere come dati di ingresso due immagini, una rappresentante l'immagine campione, da cui prendere spunto di volta in volta, e l'altra consistente nell'immagine affetta da rumore, da trasformare in modo da ottenere una nuova immagine con caratteristiche simili all'immagine campionaria.
Chiameremo qui, per semplicità, l'immagine campionaria immagine sorgente e l'immagine rumorosa, immagine destinazione.

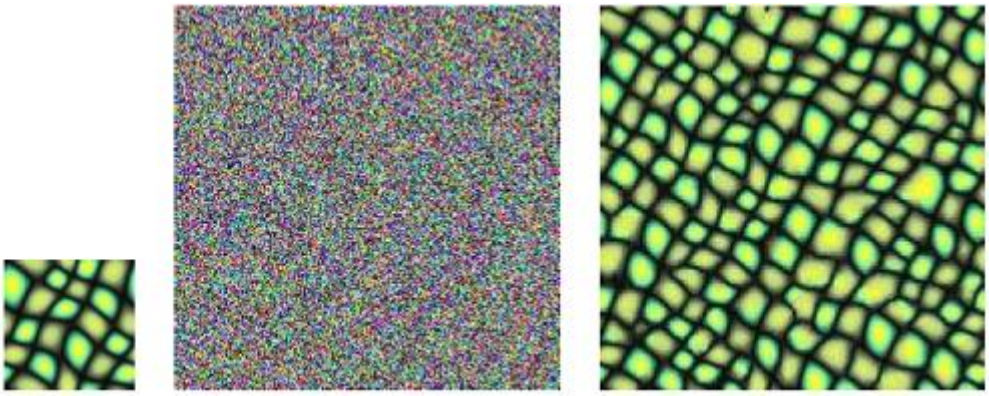


Fig. 10.4 - Immagine sorgnete a sinistra, immagine destinazione al centro,
immagine destinazione dopo l'esecuzione dell'algoritmo di sintesi della tessitura, a destra

In parole povero ciò che l'algoritmo esegue, consiste in una ricerca esaustiva.

Partendo dal pixel in alto a sinistra dell'immagine destinazione non ancora filtrata, si cerca, per ogni pixel di questa immagine, il pixel che più "somigli" a quello in esame e che appartenga all'immagine sorgente. Questa simiglianza è ottenuta tramite ben definite misure che vedremo successivamente.
Una volta trovato il pixel appartenente all'immagine campionaria, ideale per una corretta sostituzione nell'immagine di destinazione, alla posizione indicata, se ne effettua la copia e si passa al pixel successivo.
La visita dei pixel indagati sull'immagine rumorosa procede pedissequamente per righe e colonne, dal pixel in alto a sinistra a quello in basso a destra.
Nel resaturo digitale le variazioni sono minime.
Innanzitutto non si hanno due immagini in ingresso, ma una sola, sulla quale è presente sia l'immagine sorgente che l'immagine destinazione.

Fig. 10.5 - Nella texture synthesis applicata al restauro digitale si hanno immagine sorgente (area nel quadrato in nero) e immagine destinazione (tutta l'area rossa) racchiuse all'interno della stessa immagina

Possiamo notare dalla figura precedente quanto appena affermato. L'immagine affetta dal danno nel resturo digitale, contiene sia aree di sorgente che di destinazione per algoritmi di sintesi della tessitura.
La zona coperta dalla maschera, l'ormai nota Ω, è la destinazione, l'area su cui effettuare la sostituzione dei pixel in modo da restaurarli, mentre tutto il resto dell'immagine (anche se poi in realtà ne verrà utilizzata solo una porzione,

prossima dal difetto da restaurare come indicato in figura) corrisponde all'area sorgente, da cui attingere dati utili per la copia nella zona difettosa.
Notiamo immediamente come nessun nuovo valore di colore viene creato, come invece avveniva negli algoritmi [7] e [9], dove, attraverso meccanismi di diffusione dei valori di colore dei pixel attorno a quello in esame, si calcolava un nuovo pixel. Qui invece i pixel (e quindi i colori) sono direttamente copiati, sostituendo i pixel del difetto con valori già presenti nell'immagine.

Fast Texture Synthesis using Tree-Structured Vector Quantization

L'algoritmo [10] in se, proposto nel 2000 da L.Wei e M.Levoy, è nato per utilizzo in texture synthesis pura. Gli input di cui necessità sono, come già spiegato, una texture campionaria e un'immagine rumorosa da trasformare in modo da darle un aspetto simile a quella dell'immagine campione.
La texture sintetizzata, che costituisce l'output dell'algoritmo, può avere dimensioni arbitrarie, indipendentemente dalle dimensioni della texture compionaria.
L'algoritmo inizia dal primo pixel in alto a sinistra dell'immagine destinazione in ingresso, non ancora restaurata, e ne considera la matrice di pixel vicini a quello in esame. Questa matrice viene copiata e se ne cerca una sua simile lungo tutta la superficie dell'immagine sorgente, attraverso un'apposita misura di similitudine.
Una volta ricavato il best match, si copia il pixel centrale della matrice, preso ovviamente dalla texture d'origine, al posto del pixel, sempre nella posizione centrale della matrice, nell'immagine affetta da rumore.

Cerchiamo di essere più chiari utilizzando qualche esempio.
Consideriamo l'immagine rumorosa di partenza

Fig. 10.6 - Il primo pixel non possiede nessun vicino utile alla creazione della matrice per la ricerca lungo l'"immagine sorgente, risiedono tutti al di fuori dell'immagine destinazione

notiamo che, come già accennato, l'algoritmo inizia dal primo pixel in alto a sinistra, pur non avendo questo alcun vicino "utile" per creare la matrice, chiamata "textons", che servirà alla ricerca del suo simile lungo la texture campionaria.
La ricerca di questo pixel diviene pressochè casuale all'interno della tessitura d'origine.
Il numero di vicini aumenta man mano che si procede con l'algoritmo, sino a raggiungere la capienza massima dovuta alle dimensioni scelte della matrice.
Come possiamo vedere dalle seguenti figure

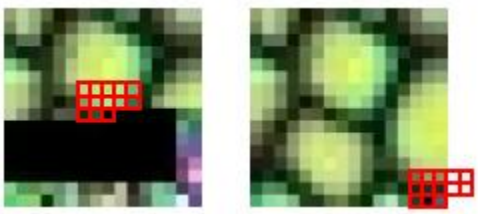


Fig. 10.7 - A sinistra il pixel indagato è nel mezzo dell'immagine rumorosa, la sua textons ha la mssima capienza, a destra si sta operando sull'ultimo pixel dell'immagine di destinazione

dove l'ultima immagine a destra ha eseguito l'algoritmo sull'ultimo punto affetto da rumore, terminando l'esecuzione.
Dalla seguente figura

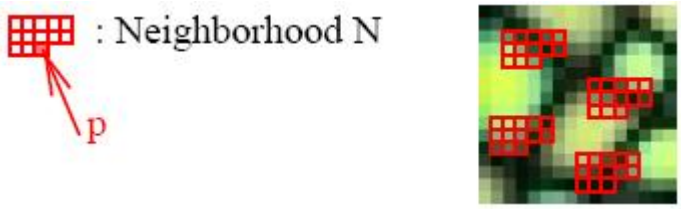


Fig.10.8

osserviamo la texture campionaria (sulla destra) con alcune delle scelte di "best match" operate durante l'algoritmo.
Alla sinistra della figura 10.8 notiamo invece la matrice di ricerca, in cui viene indicata la posizione del pixel in esame relativamente alla matrice stessa.
Possiamo vedere da tutte queste figure come la matrice non abbia la classica forma quadrata, ma bensì una struttura a L, questo è dovuto proprio alla metodologia di ricerca che si sposta da sinistra verso destra e dall'alto verso il basso lungo l'immagine di destinazione. Quindi i pixel "successivi" a quello in esame sono ancora affetti da rumore e quindi non utilizzabili nella creazione della textons.
La ricerca si baserà dunque sui soli pixel che "precedono" quello attualmente indagato.
Proprio per questa caratteristica il primo pixel è scelto pressochè casualmente, non possedendo nessun vicino valido per il confronto. Quelli immediatamente adiacenti sono scelti in maniera meno efficiente rispetto a quelli posizionati nella parte centrale dell'immagine, possedendo meno vicini utilizzabili rispetto a questi.

La misura di similitudine utilizzata nella ricerca, lungo la texture campionaria, di un corretto sostituto da copiare nell'immagine di destinazione, è la semplice *SSD* (Sum of Squared Differences).
Consideriamo N_1 e N_2, le due matrici corrispondenti alle due immagini in ingresso, la similitudine tra loro è calcolata come

$$D(N_1, N_2) = \sum_{p \in N} [N_1(p) - N_2(p)]^2$$

Se N_1 è la matrice dell'immagine di destinazione e N_2 quella posizionata sulla texture campionaria, ci si accorge che durante la computazione della $D(N_1, N_2)$, la matrice N_1 rimane "ferma", è sempre posizionata in modo da avere come pixel centrale quello da sostituire, sull'immagine destinazione.
Solamente N_2 cambia il suo contenuto, in quanto verrà spostata attraverso tutta l'immagine sorgente, ricercando quei valori di colore dei pixel che, inseriti nella nostra N_2, rendono minima l' *SSD* rispetto a qualsiasi altra possibile combinazione di pixel in N_2. Terminata la visita dell'immagine campionaria, dopo aver confrontato tutte le possibili N_2 con la N_1 dell'immagine di destinazione, avremo salvato da qualche parte la N_2 per cui la D sia minima e le coordinate del pixel centrale relativo, in modo da poterlo copiare al centro di N_1.
La dimensione delle matrici N_1 e N_2 (che ovviamente deve essere la stessa) è una scelta arbitraria fatta dall'utente e dovrebbe dipendere dalle dimensioni del più piccolo dettaglio presente nell'immagine.
Grandezze tipiche sono 5x5, 7x7, 9x9 e 11x11. Più è grande la matrice di ricerca e maggiore sarà la probabilità di trovare pixel adeguati ad essere posizionati nel punto richiesto durante la ricerca, ma allo stesso tempo aumenterà anche la complessità temporale.

E' comunque ben chiaro che più un'immagine è dettagliata e maggiori dovranno essere le dimensioni della matrice di ricerca.
Il metodo analizzato è a singola risoluzione.
Nello stesso articolo [10] è proposta una variante di questo algoritmo, in modo che operi su una *piramide Gaussiana* generata a partire dalle solite due immagini di input.
Una caratteristica delle immagini digitali risiede nell'alta correlazione che lega pixel tra loro vicini. Rappresentare quindi le immagini usando il valore di ogni singolo pixel è un'operazione altamente inefficiente: molte informazioni sarebbero ridondanti. Basandosi su questa evidenza sperimentale Adelson et al. proposero in [11] una nuova rappresentazione delle immagini detta appunto piramidale.
Introduciamo il concetto di piramide Gaussiana, partendo da un esempio

Fig. 10.9 - Immagine e sua piramide gaussiana

Il termine piramide sta a indicare la costruzione di una serie di immagini a risoluzioni via via decrescenti partendo da un'immagine iniziale.
Precedentemente si è parlato di convoluzione tra i pixel dell'immagine e alcuni kernel Gaussiani, usata, in quel contesto, per ottenere la diffusione isotropa.
La costruzione di una piramide gaussiana è un procedimento molto simile.
Il primo passo dell'algoritmo per la creazione di tale piramide consiste nel filtrare l'immagine originaria, che indichiamo per semplicità con g_0, tramite un filtro passa basso che apporti una diminuzione di risoluzione di un fattore due, ottenendo così una nuova immagine, g_1, chiamata immagine ridotta di g_0.
L'algoritmo opera ricorsivamente su g_1, in maniera analoga, apportandovi un'ulteriore riduzione di risoluzione, e arrestandosi solo quando g_k è formata da un unico pixel. La piramide così ottenuta rappresenterà tutti i possibili livelli realizzabili partendo dall'immagine a disposizione. Noi non avremo bisogno di operare con tutti quanti i livelli, ma ci accontenteremo di averne due o tre.
Il caso più semplice per la creazione della piramide, calcola la media di quattro pixel adiacenti e assegna a ognuno dei quattro pixel questo nuovo valore medio, ottenendo una nuova immagine con risoluzione inferiore all'originale.

Fig. 10.10 - L'immagine a sinistra è l'originale, man mano che si va verso sinistra si perde in risoluzione di un fattore due

Successivamente ad una riduzione di questo tipo, molti pixel adiacenti avranno lo stesso identico valore (ogni 4 pixel al primo livello di riduzione, ogni 16 al

secondo, …) è evidente la quantità di informazione ridondante.

Fig. 10.11 - Osserviamo a sinistra un blocco di pixel preso dall'immagine originale. Al centro lo stesso blocco dopo la prima diminuzione di risoluzione. A destra sempre lo stesso blocco dopo un'ulteriore diminuzione.

Per questo motivo vengono diminuite le dimensioni delle nuove immagini, in modo da compattare in unico pixel tutti quelli che posseggono lo stesso valore medio e sono adiacenti.

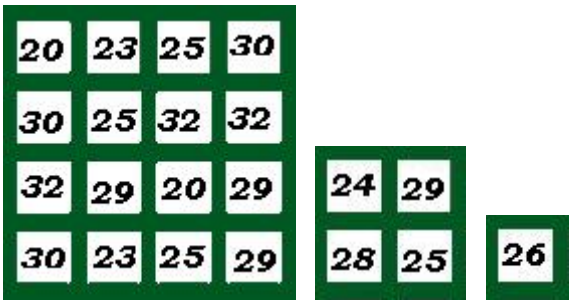


Fig. 10.12 - Lo stesso blocco precedente. Sono state ridotte le dimensioni dei blocchi a minor rioluzione, scartando inutili informazioni ridondanti

Fig. 10.13 - Le stesse immagini di prima ridimensionate

Si noti che la figura a sinistra è l'originale, quella centrale ha dimensioni pari alla metà dell'originale e quella a destra un quarto della prima e la metà della centrale.

Questa operazione potrebbe sembrare irreversibile, ricostruire l'immagine orginale è impossibile proprio per via di questa mediatura sul valore dei pixel che rende impraticabile il recupero dei valori originali.

In realtà siamo in grado di mantenere i dati che vengono perduti nel passaggio di dimuzione di risoluzione, permettendoci, in fase di ricostruzione, di ricreare l'immagine originale.

Per evidenziare i dettagli che vengono eliminati in ogni fase della costruzione della piramide si usa la *piramide Laplaciana*, simile alla Gaussiana ma ottenuta sottraendo all'immagine originale, l'immagine data dal processo di sfocamento.

Fig. 10.14 - Immagine Laplaciana. Immagine Originale. Immagine Gaussiana

In questo modo è evidente come sia sempre possibile riottenere l'immagine di partenza dalla somma dell'immagine Laplaciana e della Gaussiana.

Quando l'algoritmo di sintesi della tessitura a singola risoluzione è applicato ad una piramide si ottiene la multi-risoluzione.
Pensiamo di avere la nostra immagine originale in ingresso e di creare, a partire da questa, una piramide con al massimo tre livelli. Indichiamo con G_0 il primo livello, consistente nell'immagine originale, G_1 il secondo livello e G_2 il terzo livello.
Dalla differenza tra l'immagine originale e G_1 otteniamo il primo livello della piramide Laplaciana, L_1, mentre sottraendo a G_1 l'immagine G_2 se ne costruisce il secondo livello, L_2.
L'algoritmo inizia a lavorare proprio sull'ultimo livello della piramide Gaussiana, G_2, applicandovi la procedure a singola risoluzione precedentemente esaminata.
Terminata l'esecuzione su G_2, che ora non sarà più affetta da rumore ma avrà caratteristiche simili alla texture campionaria, bisogna ricreare G_1.
L'operazione di ricreazione è eseguita in modo opposto a quella di diminuzione di risoluzione, anziché comprimere quattro pixel adiacenti in uno solo, dimezzando le dimensioni totali dell'immagine, si effettua l'espansione di un pixel, in modo da ottenere un G_2 con le dimensioni di G_1. Dopo questa espansione viene effettivamente ricreato G_1 dalla relazione $G_1 = G_2 + L_2$.
Adesso l'algoritmo a singola risoluzione deve essere applicato a G_1 ed è proprio qui la differenza sostanziale. G_1 ora non è un'immagine completamente rumorosa, ma è una immagine ottenuta dall'algoritmo di texture synthesis al livello inferiore. Possiede quindi dei dati aggiuntivi che sono utilizzabili per effettuare la ricerca sull'immagine campionaria e che G_2 non aveva a disposizione. L'area coperta dalla maschera è stata restaurata in G_2, quindi ora non contiene più il difetto, ma il risultato del restauro su G_2.

La matrice di ricerca, la textons, non dovrà più essere necessariamente a forma di L per via della mancanza di pixel successivi a quello in esame, ma potrà benissimo avere una forma quadrata, contenendo più caratteri di confronto e permettendo un miglior risultato nel calcolo della similitudine.
Tutte le idee e considerazioni si qui espresse si possono trasferire al campo del restauro digitale, tenendo sempre conto della differenze basilari, come la presenza di immagine sorgente e immagine destinazione all'interno della stessa immagine fornita in ingresso.
Inoltre, poiché non si opera per righe e colonne in modo lineare, ma ci si sposta lungo il contorno della maschera, la matrice di ricerca non avrà una forma prestabilità, ma dipenderà dal livello di "frastagliamento" del contorno stesso.

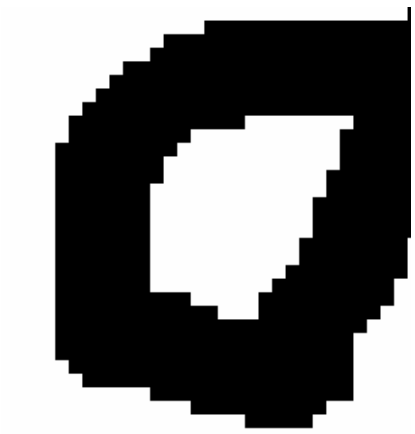


Fig.10.15 – Particolare di una maschera. E' evidente come questa sia non regolare sulla maggior parte dei pixel di contorno

Solo nei casi in cui il difetto sia perfettamente livellato (estremamente rari, soprattutto se si pensa che la maschera è realizzata manualmente dall'utente) potremo permetterci una textons perfettamente a L (se si opera sul livello più basso, altrimenti una matrice perfettamente quadrata).

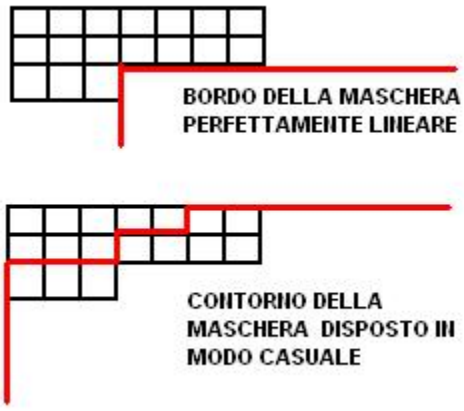


Fig.10.16

Nell'immagine possiamo notare i due casi appena citati, il primo di contorno della maschera (in rosso) perfettamente lineare e matrice a forma di L che vi si adagia combaciandovi e il secondo, il caso usuale, di bordo della maschera irregolare. Inoltre nemmeno la textons lungo l'immagine campionaria potrà sempre essere di forme ben definite nel restauro. Questa è una caratteristica in parte dovuta alla convivenza nella stessa immagine di area sorgente e area destinazione (Fig. 10.17). Ricordiamo infatti che la zona di ricerca (sorgente) comprende un'area di dimensioni massime 11x11 centrata nel pixel indagato. I pixel più prossimi a quello in esame siti anch'essi nei pressi della maschera, ne saranno coinvolti per forza di cose.

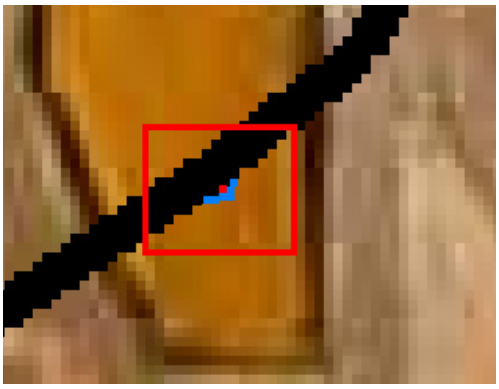


Fig. 10.17 – Particolare di un'immagine. In rosso il pixel indagato. All'interno del quadrato rosso si ha l'area di ricerca del sostituto. In azzurro alcuni possibili pixel "sostituti", anche questi, come il pixel indagato, possiedono alcuni vicini coperti dalla maschera

Per ovviare a questo inconviente si è pensato ad una semplice tecnica: innanzitutto la matrice non sarà a forma a di L, bensì una matrice prettamente quadrata, indipendentemente dal livello piramidale. Nel calcolo della *SSD* verranno poi presi in cosiderazione solamente i pixel che non sono coperti dalla maschera in entrambe le matrici, ovvero quelli che sono non affetti dal danno o che sono già stati restaurati, sia che essi appartengo a N_1 sia che facciano parte di N_2, in modo da poter operare sugli stessi pixel validi nelle due. Basta che un pixel sia "nero" in una delle due matrici per non essere preso in considerazione.

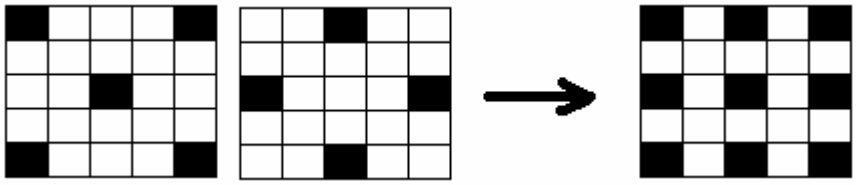


Fig 10.18 - In nero abbiamo i pixel coperti dalla maschera. La prima matrice rappresenta N_1, mentre la seconda N_2. La terza matrice indica i pixel che saranno utilizzati per calcolare *SSD*, ovvero quelli che non sono neri in entrambe le matrici corrispondeti

Per via di questa variabilità nel numero di pixel utilizzati per il computo della *SSD* potrebbero sorgere dei problemi: le matrici di valutazione con il maggior numero di pixel non coperti dalla maschera (quelle più lontane dal pixel indagato) avrebbero certamente l'*SSD* più alta, poiché con più termini di somma, e sarebbero certamente scartati nonostante una miglior verosimiglianza di confronto data proprio da un più alto numero di valori presenti in matrice.
Per evitare scarti errati si è pensato di dividere l'*SSD* per il numero di termini della somma

$$SSD = \frac{SSD}{\text{n° di pixel non coperti da maschera}}$$

In questo modo la somma viene “mediata” , permettendo di ottenere un computo della *SSD* relativo al numero di pixel su cui è effettivamente calcolata.
Per migliorare l’operazione sarebbe più utile cercare di restaurare per primi i pixel che possiedono il minor numero di vicini coperti da maschera, in modo da ottenere sempre più vicini già restaurati, e quindi disponibili per un eventuale confronto, con il procedere dell’algoritmo.

Questo è il concetto basilare di *priorità* che verrà esaminato nel prossimo capitolo.
In generale l’algoritmo di Wei e Levoy permette di ottenere buoni risultati di restauro in tempi davvero brevi, fornendo gradevoli continuazioni di colore, sia in zona ad alto contrasto, sia in zone uniformi, evitando gli sgradevoli sfocamenti lungo i contorni, carattestici degli algoritmi di diffusione. Alcuni casi di fallimento saranno esaminati nel prossimo capitolo, fornendo un cocetto atto alla loro eliminazione.

11. Priorità

Nel capitolo precedente abbiamo accennato al concetto di priorità parlando di una eventuale selezione dei pixel da elaborare per primi durante l'esecuzione dell'algoritmo di restauro. In quel contesto si proponeva di cominciare ad operare sui pixel con il maggior numero di vicini non coperti dalla maschera e quindi utili al computo della *SSD*. Questa è un'applicazione del concetto di priorità alquanto rozza e non apporta miglioramenti effettivi alle immagini restaurate. Tecniche nettamente superiori si basano sulla sensibilità del sistema visivo umano, in grado di sopportare eventuali artefatti in zone in cui i colori sono piuttosto uniformi, come un cielo azzurro, un mare blu, una parete bianca, portando immediatamente all'attenzione difetti localizzati in aree ad alto contrasto, come volti umani, griglie, contorni.
Proprio queste zone sono quelle in cui è necessario un migliore e più attento restauro, in modo da mantenere la continuità degli schemi strutturali dell'immagine.
Obiettivo, questo, che si propone il prossimo algoritmo in esame.

Object Removal by Exemplar-Based Inpainting

L'algoritmo, proposto nel 2003 da Criminisi et al., propone una tecnica di restauro con l'utilizzo del meccanismo della priorità appena citato.
Supponiamo di voler trattare la seguente immagine

Fig.11.1

dove la zona bianca contornata di rosso rappresenta il difetto.
Se noi operassimo utilizzando l'algoritmo [10], sul contorno della maschera, o anche semplicemente per righe e colonne, l'informazione copiata, propagata all'interno del difetto risulterebbe errata.
Il risultato dell'algoritmo di Wei e Levoy sull'immagine precedente è il seguente

Fig. 11.2 - Risultato dell'algoritmo [11] sull'immagine precedente

Notiamo come sia stata copiata informazione errata, al posto del palo si ha ora una striscia di mare, in questo modo il cartello superiore sembra volteggiare magicamento nel vuoto.
Per risolvere il problema l'algoritmo deve operare prima di tutto sulle zone ad alto contrasto (i contorni del palo) che incontrano il bordo della maschera, in modo da prolungare all'interno del difetto prima queste linee e poi le restanti zone uniformi.
In parole povere dobbiamo cercare il pixel situato lungo il contorno della maschera che presenta la più alta priorità e operare su quello.
Applicando questo concetto all'immagine precedente si ottiene

Fig.11.3

Il miglioramento è evidente, il palo è continuato e nessun artefatto è creato dall'algoritmo di restauro.
Analizziamo ora in dettaglio il concetto priorità.
Rimane valida la notazione utilizzata sino a qui e introdotta durante l'esposizione dell'algoritmo "Image Inpainting", riassunta nella figura 11.4

Fig. 11.4 - Ω è l'area coperta dalla maschera, $\partial\Omega$ il fronte di riempimento, Φ è la regione sorgente in cui cercare i pixel "sostituti"

Φ, l'area di ricerca, può essere definita come tutta l'immagine tranne la parte coperta da difetto ($\Phi = I - \Omega$) anche se in realtà si utilizzerà solo una porzione di questa, composta dei pixel "prossimi" a quello in esame.

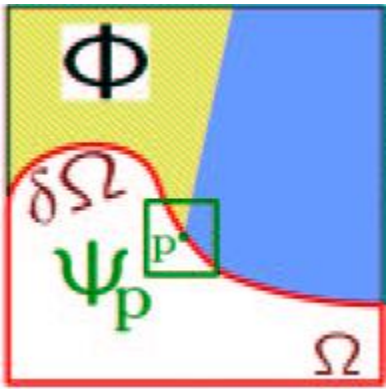


Fig. 11.5 - p è il pixel a massima priorità, ciò che vogliamo sintetizzare non è p singolo ma Ψ_p

Il pixel indicato con "p" nell'immagine precedente è il pixel a massima priorità, corrispondente ad una zona ad alto contrasto, la sua localizzazione corrisponde ad una netta variazione di colore (dal giallo al blu).
Ciò che vogliamo capire è la modalità di selezione che lo ha identificato come prioritario.
Anche questo algoritmo, come [10], opera con un procedimento "a buccia di cipolla", cioè per livelli concentrici, cercando ogni volta il pixel a massima

priorità tra quelli situati lungo il contorno della maschera.
Per ogni pixel appartenente a $\partial\Omega$ si considera la matrice Ψ_P centrata nel pixel indagato e se ne computa la priorità $P(p)$, definita dal prodotto di altri due termini:

$$P(p) = C(p)D(p)$$

$C(p)$ è chiamato "*termine di confidenza*", mentre $D(p)$ è il "*termine dato*".
La loro definizione:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \Phi} C(q)}{|\Psi_p|} \qquad D(p) = \frac{|\nabla I_p \cdot n_p|}{\alpha}$$

$|\Psi_p|$ è l'area di Ψ_p, α è un fattore di normalizzazione (α=255 per una tipica immagine a livelli di grigio) e n_p è un vettore unitario ortogonale al fronte $\partial\Omega$ nel punto p. ∇I_p rappresenta ancora il gradiente dell'immagine nel pixel p; calcolandone il modulo andiamo a cercare l'intensità del gradiente per quel pixel, più questa è alta e maggiore sarà il contrasto per quel punto.
Durante l'inizializzazione il termine $C(p)$ è settato a zero per ogni pixel all'interno del difetto, mentre $C(p) = 1$ per ogni pixel di Φ, l'area non coperta dalla maschera.
$C(p)$ corrisponde proprio a quanto introdotto nel capitolo precedente, indica cioè il numero di pixel non coperti dalla maschera che circondano il pixel in esame e sono all'interno di Ψ_p. E' una misura della quantità di informazione disponibile attorno a p. Incorporando questo termine nel calcolo della priorità si fa in modo di restaurare per primi i pixel che possiedono vicini integri.

Inoltre l'inserimento del termine di confidenza in P permette di non dover effettuare controlli aggiuntivi sulla posizione del pixel per verificare se questo sia realmente sul bordo o meno, in quanto i punti dell'immagine interni al difetto e senza vicini "scoperti" e quindi disponibili per il confronto con la regione sorgente, verranno segnalati come pixel a bassa priorità proprio grazie a C(p) e saranno riparati solo successivamente, quando altri pixel attorno saranno stati resi disponibili ai fini del confronto, poiché già restaurati.
Il termine dato $D(p)$ è una funzione delle isophote che colpiscono il fronte $\partial\Omega$ ad ogni iterazione.
Questo fattore è di fondamentale importanza nel computo dell'algoritmo perché incoraggia a sintetizzare per prima la struttura lineare presente nell'intorno del difetto, propagandola in maniera sicura all'interno di esso, in modo molto simile a quanto descritto prolissamente per l'algoritmo [7] nel capitolo 9.

Fig. 11.6 - L'evoluzione dell'algoritmo. Si può notare come sia riprodotta la struttura lineare del triangolo

Una volta trovato il pixel con la priorità massima si tiene in considerazione la matrice Ψ_p centrata nel pixel. Dimensioni tipiche di questa matrice sono 7x7 e 9x9.
Qui inizia la parte di sintesi della tessitura: si va alla ricerca di un valido sostituto all'interno della regione sorgente.
Questa area da cui attingere informazioni è composta da una matrice di pixel siti nelle vicinanze di *p*, le cui dimensioni devono essere nettamente maggiori a quelle di Ψ_p.
La ricerca è basata sull'equazione

$$\Psi_q = \arg\min_{\Psi_q \in \Phi} d(\Psi_p, \Psi_q)$$

Si calcola la distanza minima tra la pezza centrata nel pixel difettoso e una pezza, con le stesse dimensioni, presa dalla zona di ricerca.
La distanza *d* è data ancora dalla somma delle differenze quadratiche (SSD).
Una volta trovata la pezza Ψ_q che soddisfa l'algoritmo non si effettua la copia del singolo pixel come professato in [10], ma si copia l'intera pezza Ψ_q al posto di Ψ_p. Questo permette una migliore continuazione delle linee strutturali all'interno del difetto rispetto ad un algoritmo orientato al singolo pixel.
Una volta sostituita la pezza, l'algoritmo procede aggiornando entrambi i termini che compongono $P(p)$ e passando all'iterazione successiva.
L'algoritmo ottiene risultati nettamente superiori rispetto a procedimenti di inpainting tramite texture synthesis pura nel caso di zone da restaurare di dimensioni elevate, come mostra la seguente immagine,

Fig. 11.7 - Immagine originale (sinistra) e immagine modificata con l'algoritmo [12]

L'area coperta dalla donna, che viene eliminata con il procedimento descritto, è abbastanza estesa, ma il risultato finale mostrato ad un occhio ignaro dell'immagine "difettosa" non lo metterebbe nelle condizioni di notare incongruenze o artefatti, nè lungo le linee strutturali nè nelle zone uniformi. Quando invece l'area è piuttosto ristretta, come avviene in genere quando il difetto è una crepa, i risultati ottenuti sono pressochè identici a quelli ottenuti con l'utilizzo dell'algoritmo di Wei e Levoy descritto nel capitolo precedente. Questo algoritmo presenta come unica pecca la velocità di esecuzione. Durante il calcolo della priorità vanno infatti esaminati tutti i pixel che risiedono sul fronte della regione coperta dalla maschera, ogni volta. Prendiamo a paragone proprio l'algoritmo del capitolo precedente, questo visitava i pixel di contorno del difetto una unica volta, nel momento del loro restauro. Con l'algoritmo di Criminisi et al. il contorno viene visitato un numero di volte pari al numero di pixel nel difetto diviso il numero di pixel che vengono copiati di volta in volta, ovvero il numero di pixel contenuti in Ψ_p e non coperti dalla maschera. In ogni caso parecchie volte in più rispetto ad un semplice algoritmo di sintesi della tessitura.

Digital Inpainting – Survey and Multilayer Image Inpainting Algorithms

Questo algoritmo, [13], sarà introdotto in quanto utile alla comprensione dell'algoritmo successivo, ma non verrà esaminato a fondo.

Si basa su una considerazione già incontrata in [10]: l'immagine può essere suddivisa in livelli distinti e si può utilizzare, per il processo di ricostruzione, una metodologia che sfrutta la multirisoluzione.

Gli autori, analizzando un elevato numero di immagini, hanno constatato che un gran numero di esse avevano delle caratteristiche comuni, ovvero che le diverse

intensità luminose rappresentavano altrettante profondità d'immagine: a tonalità scure corrispondevano elementi vicini all'osservatore, e a tonalità chiare erano associati elementi lontani come il paesaggio di sfondo, come accade nei disegni di cultura orientale.
Questa considerazione ha permesso di suddividere l'immagine in più livelli distinti ognuno dei quali contiene una porzione dell'immagine originale corrispondente ad una determinata luminosità e di conseguenza un determinato piano focale, come si può vedere dalla figura

Fig. 11.8 - Immagine originale e sua divisione in due livelli

Ciascuno di questi livelli viene poi ricostruito utilizzando una tecnica basata sulla multirisoluzione.
In pratica, ciascuna zona danneggiata, segnalata su ciascun livello attraverso la solita maschera disegnata manualmente, viene suddivisa in blocchi di uguale dimensione e ciascun blocco viene analizzato sotto l'aspetto della varianza dei pixel in esso compresi; se la varianza dei punti è alta il blocco in esame comprenderà una zona dell'immagine molto dettagliata, mentre se la varianza è bassa l'area compresa sarà piuttosto omogenea.
Nel primo caso si divide ulteriormente il blocco in altri sottoblocchi, come illustrato nella figura seguente, che verranno nuovamente analizzati al fine di determinare zone omogenee. Questo consente di preservare eventuali dettagli presenti sul bordo della zona danneggiata e ricostruirli all'interno dell'area danneggiata. Viceversa, se il valore della varianza è basso, si ripristinano i punti

segnalati come danneggiati all'interno del blocco con il colore medio dei pixel non danneggiati.

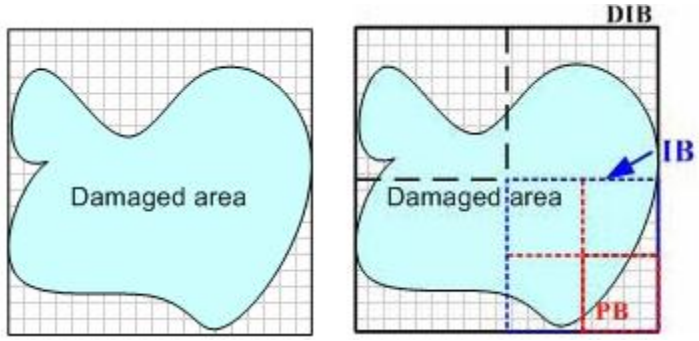


Fig. 11.9 - Viene mostrata la divisione dell'immagine in sottoblocchi

Nella figura si può vedere una schematizzazione del meccanismo di suddivisione dell'area danneggiata: l'area definita con DIB è la zona affetta dal difetto e delimitata dal blocco principale, l'area IB è stata costruita da una prima suddivisione del blocco principale in quanto, essendo un'area molto vasta, sarà presumibilmente determinata da una varianza piuttosto elevata; infine l'area PB è l'area a bassa varianza utilizzata per il computo della media dei valori dei punti in essa compresi da attribuire ai pixel danneggiati appartenenti allo stesso sottoblocco.
Una volta che ciascun livello è stato ricostruito si procede alla loro riunificazione per ripristinare l'immagine. Per fare ciò si segue una strategia precisa: con la suddivisione dei vari livelli si è già stabilito quale di essi debba essere inserito per primo e quale per ultimo, secondo il criterio che i livelli dello sfondo devono stare dietro i livelli di primo piano; durante il processo di inpainting, però, si è assegnato a ciascun pixel, oltre al valore di colore, anche un termine indicante la distanza dal bordo dell'area danneggiata e quindi l'affidabilità dell'informazione riportata; in questo caso per ogni pixel che deve essere inserito nell'immagine finale viene fatta una comparazione tra livelli adiacenti per vedere quale tra i pixel appartenenti a livelli distinti, ma coincidenti come posizione, deve essere

preso in considerazione. Verrà scelto quello che avrà una distanza inferiore poiché ritenuto più attendibile in quanto ripristinato da una zona integra più vicina. Questa distanza è stata introdotta per il fatto che dovendo dividere l'immagine in livelli ci saranno alcuni di essi che conterranno una maggior parte di pixel danneggiati rispetto ad altri e quindi che saranno stati ripristinati in modo meno affidabile.
Questo tipo di algoritmo, ha molte limitazioni in quanto può funzionare bene nel caso in cui le immagini rispecchino le caratteristiche descritte, ma presenta inconvenienti nel caso in cui i presupposti vengano invertiti come ad esempio nel caso in cui si usi una fotografia con soggetto in primo piano chiaro su sfondo scuro; gli stessi autori ammettono che l'algoritmo da loro implementato è complicato da usare nelle immagini di concezione occidentale in quanto sono molto più ricche dal punto di vista cromatico e meno soggette a costrizioni culturali per quel che riguarda gli schemi di luminosità.

Un altro algoritmo

Il prossimo algoritmo in considerazione, [15], è stato realizzato e proposto da Marco Ortolan uno studente del corso di laurea magistrale in ingegneria elettronica presso l'Università degli Studi di Trieste nel suo lavoro di tesi durante l'anno accademico 2004-2005.
Questo algoritmo è studiato appositamente per la rimozione di crepe tramite restauro virtuale di materiale fotografico deteriorato, per lo più stampe antiche, in bianco e nero. Non sono state fornite descrizioni dell'applicazione su fotografie a colori. Proporremo più avanti un semplice metodo per rendere l'algoritmo funzionante anche sulle immagini colorate.

L'algoritmo prende spunto da alcune delle tecniche precedentemente descritte, combinandole insieme. La procedura utilizza, in un primo momento, un sistema di "tassellamento", molto simile al lavoro eseguito dagli artigiani che lavorano sui mosaici, cercando, in zone vicine al difetto da ripristinare, dei pixel che si possono utilizzare per essere copiati in queste zone rovinate al fine di poter continuare la struttura dell'immagine; utilizzando gli elementi dell'immagine, ovvero utilizzando pixel aventi la giusta intensità luminosa, si evita di creare valori che mal si integrano col contesto generale dell'immagine.
Per condizioni particolari dell'immagine, infatti, dovute a una ingente usura del supporto, è possibile che non si riesca a reperire informazioni utili per una corretta sostituzione ed in questo caso si lascerà il compito di ripristino di tali punti ad un ulteriore algoritmo, un metodo d'inpainting che cerca di minimizzare le variazioni cromatiche tra i punti integri e quelli da sostituire in modo da non creare discrepanze troppo marcate tra le due zone dell'immagine.
Questo meccanismo viene effettuato attraverso un algoritmo d'inpainting simile a quello descritto in [13], ma che tiene maggiormente in considerazione l'alta attività della zona in esame ed inoltre, sfruttando la maggiore informazione proveniente dalla prima fase del metodo, consente di calcolare in modo più preciso il valore dei nuovi punti, riducendo anche in questo caso l'errore sull'attribuzione dei colori.
L'inpainting tramite diffusione, per sua natura, potrebbe creare dei problemi lungo i bordi degli oggetti in quanto prendendo le informazioni circostanti il punto da ricostruire per determinarne il valore rischierebbe di confondere le intensità luminose provenienti da due oggetti adiacenti creando dei bordi sfumati.
Per evitare questo fenomeno, in modo molto simile a [9], si adottano delle barriere inserite manualmente disegnando dei segmenti lungo il bordo degli oggetti in corrispondenza dell'area che dovrà essere ripristinata.

Le barriere inserite hanno dimensione di un pixel e vanno a ricoprire parte della regione da ricostruire indicando al programma che tali punti devono essere ripristinati per ultimi.
Di seguito viene descritto in dettaglio il metodo proposto.
Possiamo suddividere l'algoritmo di restauro in tre fasi distinte, come indica la figura successiva

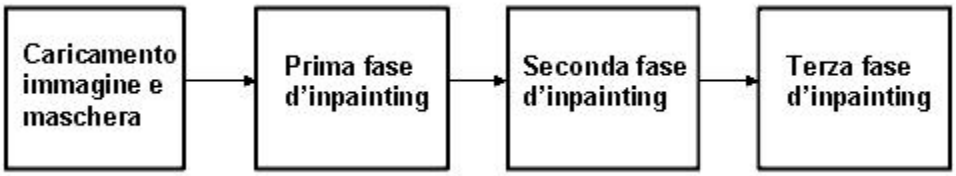


Fig. 11.10 - Macroblocchi dell'algoritmo

La fase di caricamento di immagine e maschera è comune a tutti gli algoritmi esaminati.
La prima fase inizia con l'applicazione dell'algoritmo "Fast Digital Image Inpainting", visto nel capitolo 9, sulla parte difettosa dell'immagine.
Il risultato ottenuto sarà utilizzato solamente come confronto in questa prima fase. Ne spiegheremo meglio più avanti il significato.
Dopo aver caricato le maschere ed aver eseguito l'algoritmo [9] sull'immagine, si passa all'applicazione della vera e propria "prima fase", che può essere ulteriormente scomposta nel seguente modo

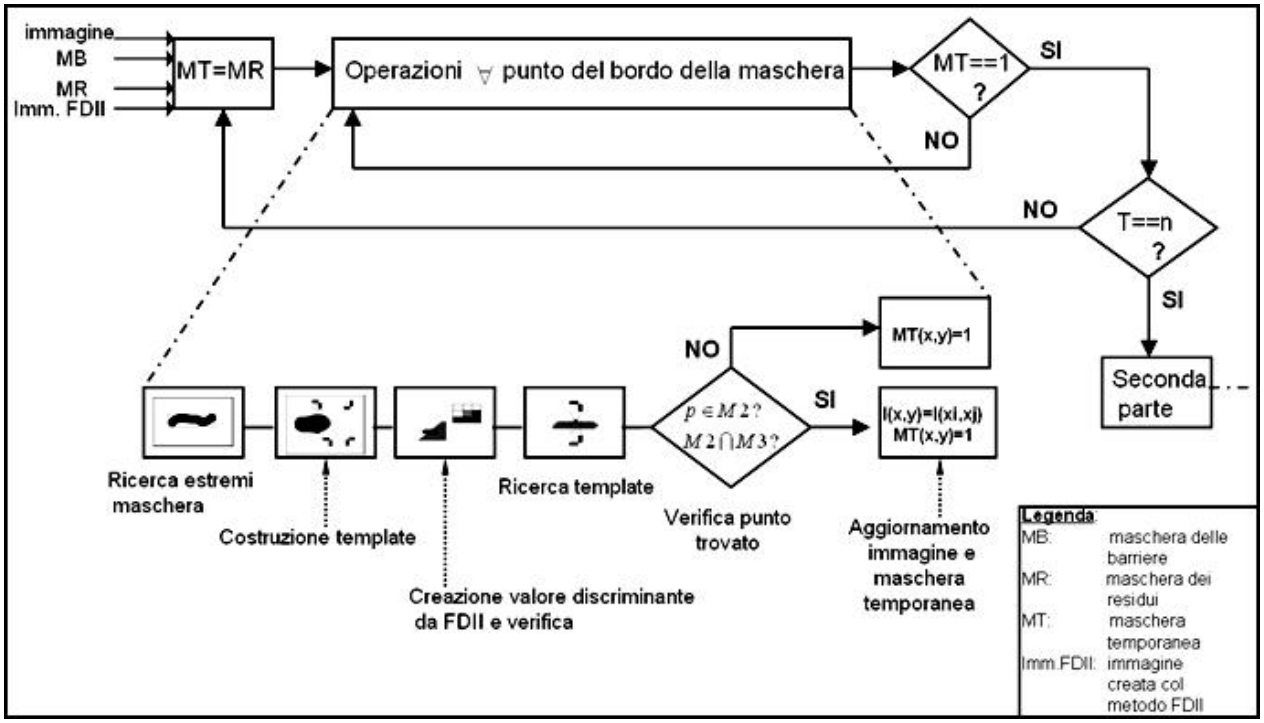


Fig. 11.10 - Schema della prima fase dell'algoritmo

Come si può notare dal diagramma a blocchi, questa parte comprende al suo interno due cicli distinti, incastonati uno nell'altro; il ciclo più interno dipende dalle condizioni di stato di una maschera, l'altro da un contatore di cicli che viene aggiornato ogni volta che il primo ciclo si sarà esaurito.
In ingresso riceve l'immagine originale, l'immagine filtrata dall'algoritmo FDII, la maschera coprente il difetto (MR) e la maschera delle barriere (MB).
Nel primo blocco del diagramma si nota che l'azione eseguita è MT=MR.
MR sta in realtà per Maschera dei Residui, mentre MT per Maschera Temporanea.
La prima maschera ha la funzione di tenere traccia di tutti quei punti che sono stati ripristinati da questa prima fase e di quelli che invece dovranno essere elaborati nella seconda parte del programma ed è proprio per questo motivo che viene denominata "maschera dei residui".

La seconda maschera invece è una maschera che viene aggiornata ogni qualvolta un punto è stato esaminato (indistintamente che esso sia stato effettivamente modificato oppure no) e proprio sullo stato di quest'ultima maschera si basa il ciclo più interno che terminerà solo quando tutti i punti della maschera saranno stati portati dal valore "0", indicante che il punto deve essere ancora valutato, al valore "255" che esprime, viceversa, che il punto è stato già analizzato; una volta che la maschera sarà stata completamente cancellata, al ciclo successivo essa prenderà il valore della maschera dei residui che è stata a sua volta modificata dal ciclo appena scaduto.
Inizia a questo punto il ciclo più interno. La prima azione consiste nella ricerca degli estremi della maschera, ovvero nel determinare quali siano le coordinate massime e minime della posizione della maschera e cioè di quei punti che devono ancora essere restaurati.
Questo passo viene effettuato ogni volta che la maschera viene modificata ed il suo fine è quello di limitare le operazioni di ricerca dei punti da esaminare entro un intorno di punti ben determinati evitando di scorrere ad ogni ciclo l'intera immagine, operazione quest'ultima di grande spreco computazionale per immagini molto grandi affette da danni limitati a regioni poco estese.
I vantaggi si possono comunque notare su ogni tipo di immagine trattata in quanto il programma prevede la cancellazione della maschera in quei punti già restaurati; quindi man mano che l'algoritmo procede la riduzione della maschera si fa più evidente, i punti da rintracciare saranno sempre meno e lo sforzo per trovare questi punti diverrà man mano più esiguo.
Nella figura seguente è presente una schematizzazione di ciò che accade durante l'esecuzione dell'algoritmo: l'area d'intervento complessiva viene sensibilmente rimpicciolita al ridursi della maschera.

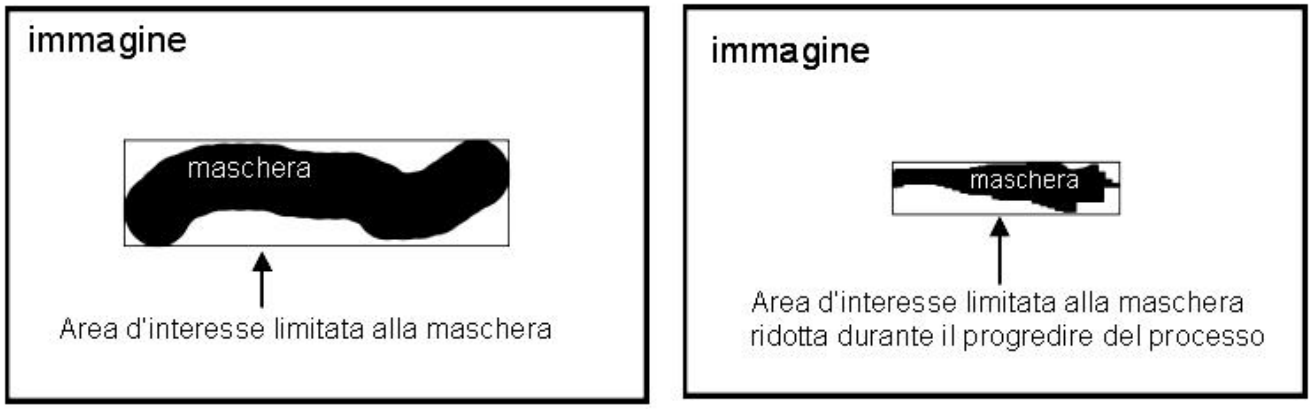


Fig.11.11

Analizziamo ora i tre blocchi soccessivi simultaneamente, in modo da dare una continuità semantica al nostro discorso, poiché costisuiscono operazioni strettamente connesse.
Nei capitoli precedenti, disquisendo di sintesi della tessitura, si descriveva la tecnica di ricerca di un pixel da sostituire ad una coperto dalla maschera, confrontando una matrice di dimensioni fissate centrata nel punto in esame con un'altra matrice, di pari dimensioni, che veniva fatta scorrere su tutta l'area di ricerca.
Questa matrice centrata nel pixel difettoso da restaurare, viene indicata nella figura con il nome di template, campione. La creazione di questo template nell'algoritmo di Ortolan è differente da quella vista precedentemente, viene infatti creato per specularità, ovvero viene "ribaltato". Questa inversione dei pixel non è fatta allo stesso modo per tutti i punti di contorno del difetto, ma dipende dal luogo in cui questi si trovano, se sopra alla maschera, di sotto o di lato.
Osserviamo l'immagine

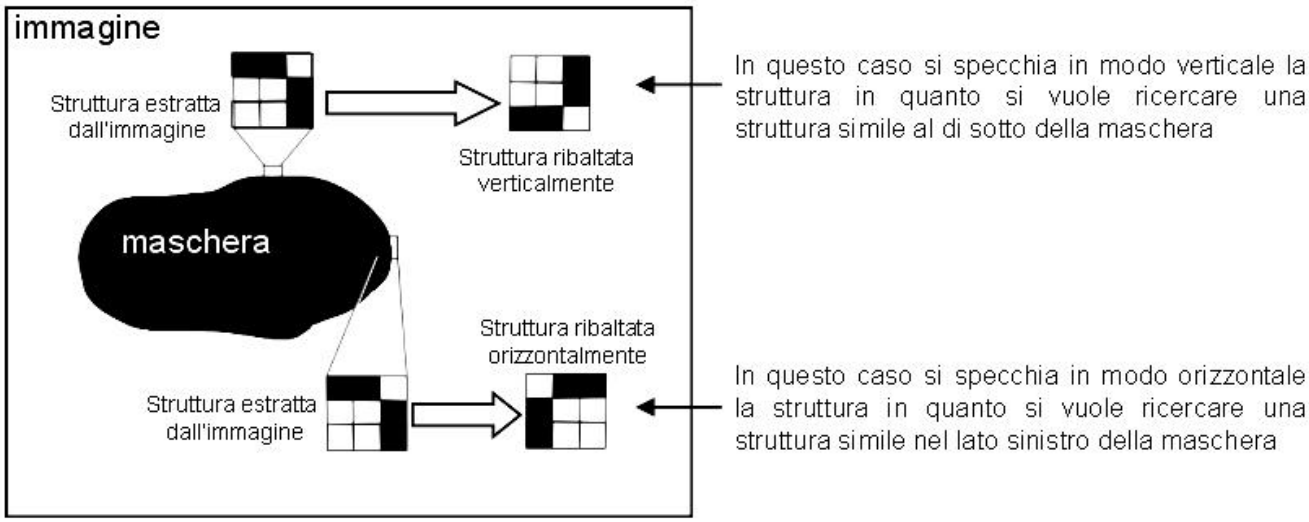


Fig. 11.12 - Schema illustrante la creazione del template e la scelta del ribaltamento

Si noti che se il pixel indagato è situato lungo la parte superiore o inferiore della maschera il ribaltamento effettuato è di tipo verticale, mentre se risiede lungo uno dei lati si effettua un ribaltamento orizzontale.
Questo ribaltamento è motivato dalla ricerca di punti simili non più nelle vicinanze dal pixel indagato, ma dalla parte opposta della maschera, in modo da dare una continuazione a strutture particolari con linee o contorni.
Una volta creato il template si dovrebbe partire con la ricerca. Ma prima viene eseguito un confronto su ogni pixel preso in esame, paragonandolo all'immagine di riferimento precedentemente ottenuta tramite l'algoritmo [9].
Si prende una matrice di dimensioni tre per tre centrata nel pixel indagato, ma posizionata sull'immagine computata da FDII

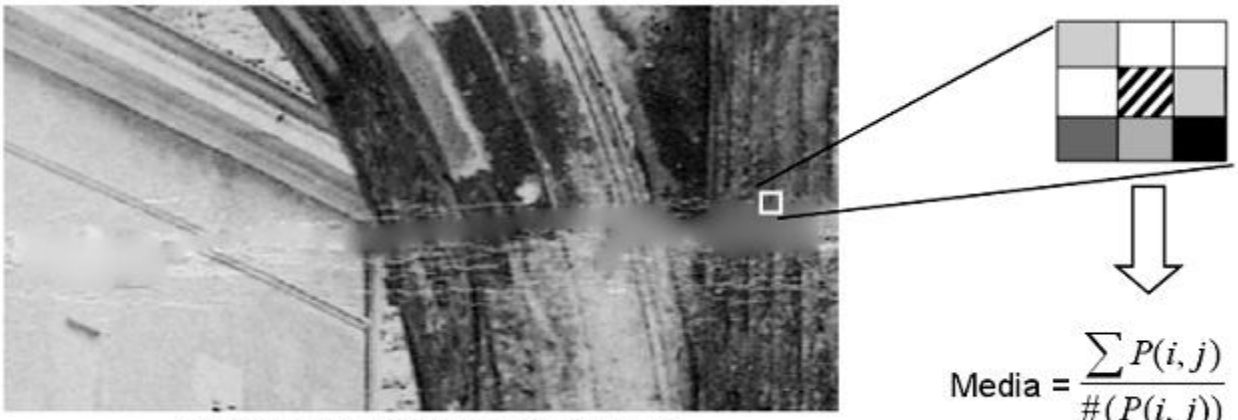


Fig.11.13

su questa matrice viene applicata l'operazione di media. Una volta ottenuta la media *m(x,y)*, il procedimento decisionale è dato dalla disequazione

$$| I(x,y) - m(x,y) | > \beta$$

dove β è una costante di soglia dipendente dalle caratteristiche dell'immagine.
Se l'espressione viene verificata, ovvero se il punto indagato si discosta troppo dal valore atteso per quella zona, allora si procede alla ricerca del punto, poiché esso è ancora danneggiato o è stato sostituito erroneamente, mentre se il suo valore è prossimo al valore atteso allora si procede con la modifica della maschera temporanea e di quella dei residui, concellandole nel punto corrispondente.
Da quanto illustrato si può capire perché sia necessario effettuare più cicli successivi e non un unico passaggio come avviene per le prossime fasi del processo di ricostruzione.
Vediamo come funziona in dettaglio la ricerca.
Anche nella ricerca, come nella creazione del template, si tiene conto della posizione relativa del pixel lungo il bordo della maschera.
La situazione è riassunta nella figura seguente.

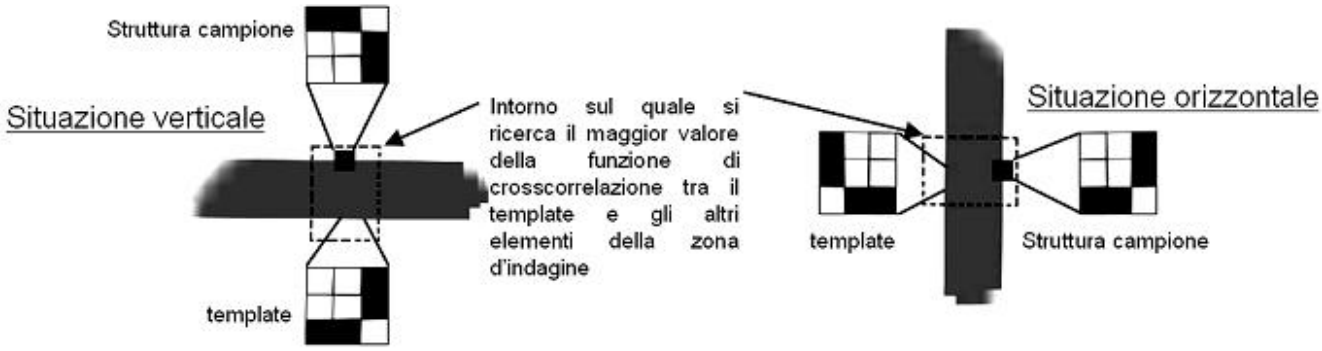


Fig.11.14

Se il pixel indagato è situato superiormente rispetto alla maschera, l'intorno di ricerca di un sostituto valido avrà una localizzazione maggiore al di sotto del difetto.
Se il pixel è nella parte inferiore si cerca sopra al difetto e così via nei casi di posizionamento laterale.
Per evitare di indagare in zone troppo lontane e quindi poco utili sia da un punto di vista computazionale sia da un punto di vista logico, in quanto si suppone che la correlazione tra due punti cali al crescere della distanza, viene presa una sottomatrice limitata di punti appartenente all'immagine nella quale si andrà alla ricerca del campione precedentemente costruito. Le dimensioni di tale sottomatrice dipendono più dalla risoluzione dell'immagine che dalla grandezza della maschera in quanto è difficile stabilire a priori quanto spessa possa essere una determinata maschera. Presumibilmente se un'immagine ha una risoluzione molto spinta si suppone che anche il difetto in essa presente sia di dimensioni elevate, parlando in termini di spessore in pixel, mentre se la sua risoluzione è piuttosto bassa anche il difetto in essa contenuto occuperà un numero di punti ristretto. Per dare comunque un'idea delle dimensioni che deve avere la zona di indagine si pensi che per trattare immagini aventi risoluzione di 300dpi (punti per pollice) si è utilizzata una matrice di dimensioni nove per ventinove punti.

Il meccanismo preposto al reperimento dell'omologo del campione prima generato nella zona da indagare è affidato alla funzione di crosscorrelazione normalizzata (FCCN) definita dalla seguente formula

$$FCCN = \frac{\sum_{x,y} campione_a(x,y) \cdot campione_b(x,y)}{\sqrt{\sum_{x,y} campione_a(x,y)^2 \cdot \sum_{x,y} campione_b(x,y)^2}}$$

dove con $campione_a$ si intende il template generato per specularità come spiegato precedentemente, mentre con $campione_b$ indichiamo la matrice di dimensioni uguali al template, che viene spostata all'interno dell'area di ricerca.
Tale funzione ha valori compresi tra zero ed uno, valori corrispondenti rispettivamente ad una totale disuguaglianza e ad una perfetta corrispondenza tra i campioni indagati; ogni qualvolta la matrice di ricerca viene traslata sulla superficie d'indagine viene calcolato il grado di somiglianza tra i due blocchi di pixel tramite la funzione di crosscorrelazione normalizzata e viene preso il valore massimo emerso. Trovato il massimo, si estrapolano le coordinate del punto centrale della matrice mobile che ha generato questo valore e che dovrà poi essere clonato nella posizione del pixel difettoso; prima di effettuare la sostituzione, però, bisogna esaminare se tale punto sia stato prelevato da un'area opportuna e permessa dell'immagine in modo da evitare errori di riempimento; nella figura che segue si ha uno schema delle zone permesse e di quelle proibite.

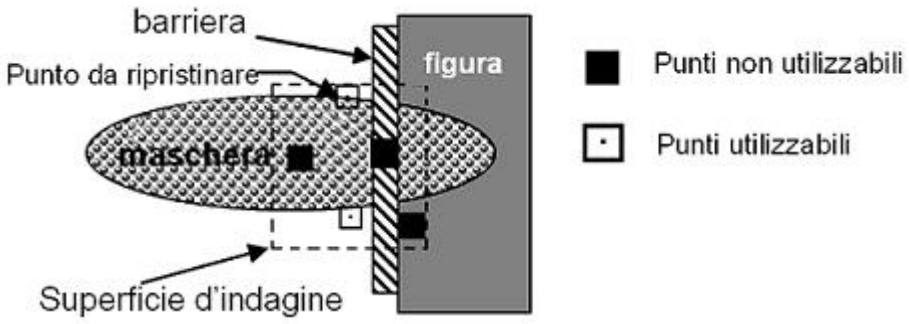


Fig.11.15

Punti non validi sono quelli che appartengono ad una delle due maschere (MT o MB)
e quei punti che stanno “al di là” di una barriera rispettivamente al pixel in esame.
Verificato che la posizione del punto rientra nelle regole prefissate, esso viene copiato nel punto da restaurare in quanto è quello che ha la probabilità massima di essere un buon sostituto. Contemporaneamente si aggiornerà la maschera temporanea. Viceversa, se il punto non viene considerato idoneo, si aggiornerà la sola maschera temporanea per indicare che il punto è stato comunque visionato.
Al ciclo successivo, tramite il metodo di confronto con l’immagine filtrata da FDII prima esposto, si verificherà se tale sostituzione è stata opportuna oppure no; in caso affermativo si lascerà il nuovo valore e si cancellerà il punto corrispondente dalla maschera dei residui, altrimenti si ricomincerà con una nuova indagine.
Nei vari cicli la struttura della maschera e dell’immagine cambia sensibilmente e per questo è necessario indagare nuovamente tutti i pixel coperti dalla maschera dei residui in quanto se inizialmente un punto non ha trovato il proprio omologo, non è detto che prima o poi non si venga a creare una situazione favorevole per una corretta sostituzione.
Termina a questo punto la prima fase dell’algoritmo.

La seconda fase necessita di ricevere come input la maschera ottenuta in output dalla prima parte dell'algoritmo, la maschera dei residui e la maschera delle barriere.
Concetto base di questa sessione è quello di priorità, similmente a quanto descritto nell'algoritmo di Shih et al. [13], dove gli stessi autori propongono di adottare come principio di priorità quello della massima varianza associabile ad un gruppo di pixel circostanti quello da rimpiazzare, in quanto sostengono che a valori elevati della varianza del colore di tali punti corrispondano zone ad elevato contrasto che sono quelle maggiormente sensibili alle valutazioni dell'occhio umano e quindi che necessitano di maggiore attenzione.
Nell'algoritmo qui proposto, per selezionare il punto d'inizio, cioè il pixel a maggiore priorità, si è deciso di prendere una matrice quadrata di dimensioni sette per sette punti contenente i valori dei pixel circostanti quello in esame che non appartengano alla zona interessata dalle due maschere, tale matrice contiene i pixel integri sui quali viene eseguito il calcolo della varianza come schematizzato in figura 11.16

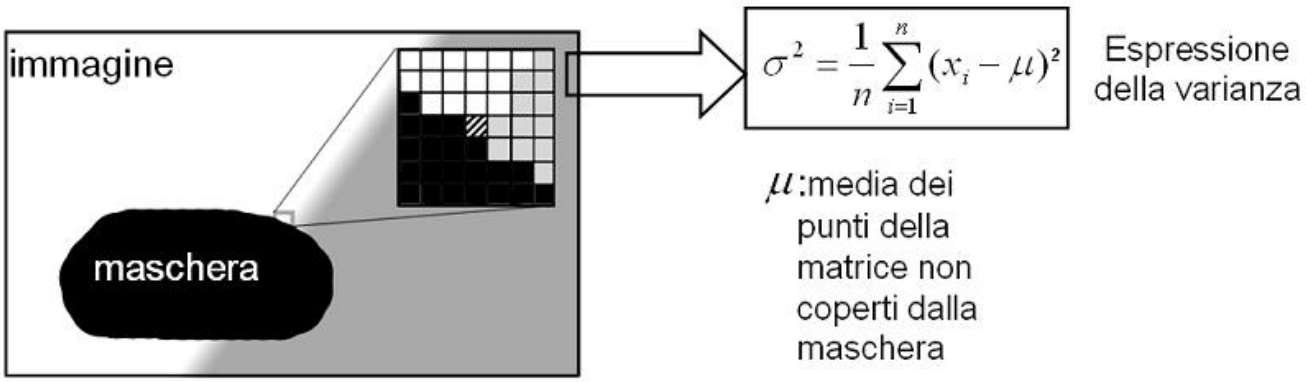


Fig.11.16

La matrice viene spostata su tutti i punti del bordo della maschera in modo che si possa stabilire quale pixel è contornato da zone a maggiore frequenza;

l'algoritmo registra per ogni spostamento della matrice gli indici del pixel a priorità relativa maggiore fino al termine della scansione, cioè quando la priorità relativa diventerà priorità assoluta. Lo schema è quindi il seguente:

$$\max(\sigma^2(x_i, y_j)) \Rightarrow p(x, y)$$

Una volta stabilita la priorità del primo pixel, questo sarà ricolorato secondo il metodo di riempimento descritto successivamente; il punto della maschera corrispondente sarà cancellato ed il processo di priorità ricomincerà daccapo considerando nuovamente tutti i pixel del bordo della maschera rimanenti fino al momento in cui tutta la maschera sarà stata eliminata.
Ciò che ora ci interessa conoscere è il metodo di riempimento, questo sfrutta la stessa struttura della ricerca della priorità, ma con un principio opposto; con questo metodo si vuole minimizzare l'energia associata al contorno del punto scelto per l'inpainting. Per fare questo si considera il valore che rende minima la varianza dei pixel considerati nella matrice che ha come elemento centrale il pixel da sostituire. Questo viene fatto per creare una uniformità tra quanto è già presente e quanto deve essere ripristinato in modo tale che non vi siano differenze visive tra le due parti e ricreare le zone corrotte il più fedelmente possibile.
Sembra ovvio che, per definizione, il valore che rende minima la varianza corrisponde al valor medio. Così non bisogna far altro che calcolare la media dei punti che circondano il pixel indagato (ovviamente solo quelli non coperti dalla maschera) e sostituire tale valore al pixel centrale.
Alcune considerazioni vanno però fatte a riguardo della parte dell'algoritmo preposta alla selezione dei valori da inserire all'interno della matrice utilizzata per il calcolo della varianza: se si prendessero tutti i valori dei pixel circostanti il punto da ripristinare, si noterebbe una sfocatura dell'area sottoposta ad inpainting in quanto all'interno della matrice potrebbero esserci indistintamente

punti di colore anche opposto (nella scala 0-255), con la conseguenza che il valore del pixel ripristinato si troverebbe ad essere sì vicino al colore dei pixel presenti in maggior numero, ma "sporcato" da alcuni valori molto diversi presenti nella matrice.
Per risolvere l'inconveniente bisogna considerare il fatto che l'immagine si può suddividere per semplicità in zone ad alta attività, ad esempio i punti che sono lungo i bordi delle figure, zone a media attività, cioè zone in cui si ha una variazione di colore, ma non intervengono cambiamenti drastici lungo la scala cromatica e zone a bassa attività, come ad esempio sfondi omogenei o che hanno delle gradazioni di colore piuttosto tenui.
Per questo si considera il valore massimo assunto dalla varianza, lungo il bordo della maschera, all'inizio della seconda fase d'inpainting. Con questo si creano tre intervalli equamente suddivisi tra il valore zero e quello calcolato e ad ognuno di essi viene associato un determinato metodo per costruire i campioni dai quali verrà calcolato il colore da assegnare al pixel da sostituire.
Una volta scelto il punto con il metodo della priorità il valore della varianza ad esso associato determinerà quale metodo utilizzare per il punto specifico.
Per le zone ad alta varianza, si considerano i pixel, come detto precedentemente, si calcola il loro valore medio e si contano quanti pixel sono inferiori a tale valore (pixel scuri) e quanti sono ad esso maggiori (pixel chiari); se si ha una prevalenza di punti scuri quelli chiari vengono sostituiti con il massimo valore dell'insieme di maggioranza, se si ha una prevalenza di punti chiari i valori inferiori alla media vengono sostituiti con il minimo valore appartenente all'insieme che ha il maggior numero di elementi. Questa sostituzione è utile in quelle zone vicine ai bordi degli oggetti per evitare che le intensità luminose di oggetto e sfondo vengano mescolate si opera come spiegato.
Per le zone identificate da valore della varianza intermedio viene utilizzato un altro metodo simile al precedente che sostituisce il valore dei punti in minoranza, indistintamente da quali essi siano, con il valore medio

precedentemente calcolato per separare i due gruppi.
Per le regioni dell'immagine caratterizzate da attività molto bassa non si applica alcun tipo di sostituzione dei valori all'interno della matrice di selezione dei punti in quanto, in questo caso, i valori dei punti sono piuttosto simili tra loro. Utilizzare direttamente il valore medio su di essi accresce l'omogeneità dell'immagine in queste regioni; per di più, se si utilizzasse uno dei precedenti metodi di sostituzione si potrebbero creare delle aree a blocchi poco piacevoli per l'osservatore.
Il processo viene eseguito fino a quando tutti i punti della maschera principale non saranno stati completamente cancellati. Terminata questa seconda fase, con la maschera dei residui interamente cancellata, rimangono da analizzare e restaurare solamente i pixel coperti dalla maschera delle barriere. Il procedimento è identico a quello appena analizzato.
La sola differenza è che, essendo la regione d'interesse larga solo un pixel, non ha senso ricercare ogni volta che è stato ripristinato un punto il pixel successivo, in quanto la struttura dell'immagine non sarà modificata in modo sensibile nelle zone appena ricostruite. Per tale motivo per risparmiare inutili operazioni si esegue un'unica scansione dei punti memorizzando per ognuno di essi le coordinate ed il valore della varianza corrispondenti in un registro. I punti vengono poi presi successivamente rispettando l'ordine decrescente del valore della varianza registrati. L'algoritmo ottiene ottimi risultati su aree di difetto relativamente piccole, tenendo conto che è stato studiato appositamente per il restauro di crepe, il cui spessore non è mai esagerato. Appena la zona si allarga l'algoritmo tende a fallire, cadendo nei difetti di diffusione dovuti alla sostituzione del pixel danneggiato con la media dei suoi vicini. Il tempo di esecuzione non è dei migliori, ancora una volta per via della ricerca esaustiva del pixel a priorità massima, che obbliga a visitare l'intera area ricorperta dalla maschera per ogni pixel in essa inclusa.

12. Scomposizione dell'immagine

Ogni immagine è composta da zone differenti, alcune maggiormente ricche di dettagli, altre più uniformi. Mentre queste ultime se restaurate con algoritmi di semplice diffusione dell'informazione, come avviene in [9], forniscono ottimi risultati, le aree dettagliate necessitano una trattazione minuziosa. Errori lungo un contorno sono nettamente più sensibili all'occhio umano, in questi punti le sfocature sono poco accettabili. Per questo le tecniche di diffusione in tali aree sono poco consigliabili. L'idea qui introdotta si basa sui concetti appena evidenziati: anziché restaurare l'intera immagine con un un unico metodo, si divide l'immagine nelle sue componenti e vi si applicano differenti algoritmi di inpainting.

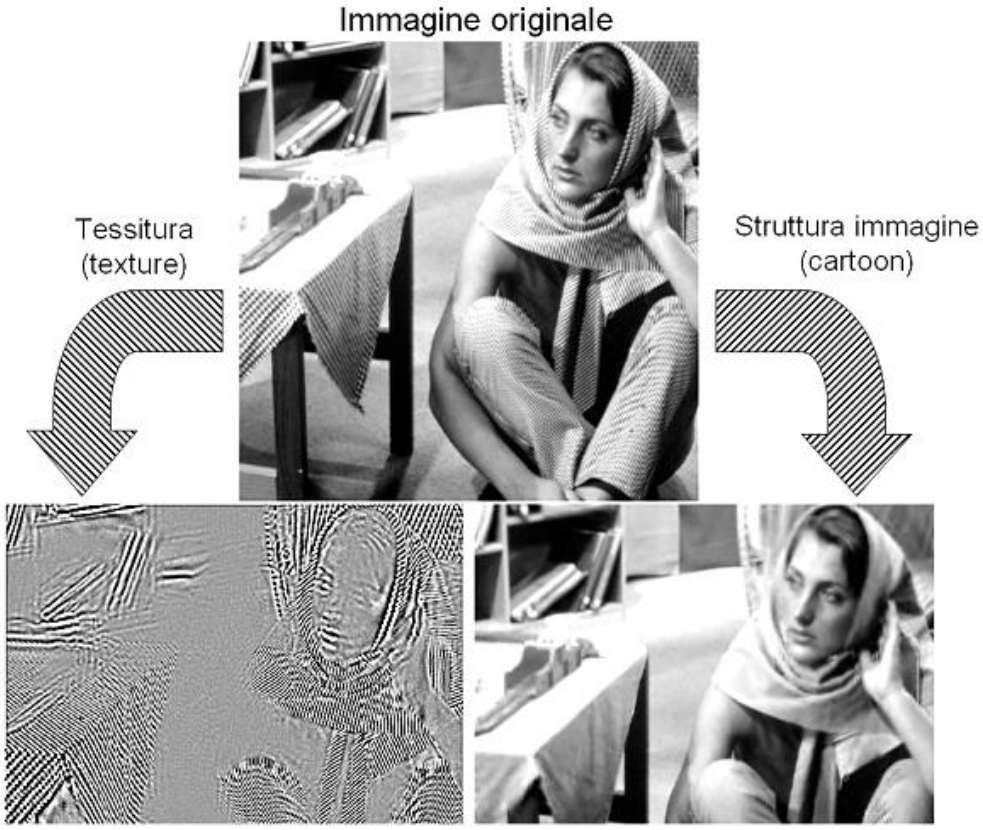


Fig. 12.1 - Scomposizione della figura nelle sue parti informative fondamentali: tessitura e struttura

Image Restoration Using Multiresolution Texture Synthesis and Image Inpainting

L'algoritmo che andremo ora a esaminare [17] è stato proposto nel 2003 da H.Yamauchi et al. e utilizza proprio la tecnica di scomposizione dell'immagine appena citata, combinando procedure di sintesi della tessitura con inpainting tramite diffusione.

L'idea di base è racchiusa in modo esplicativo nell'immagine precedente.

Si scompone l'immagine originale nelle sue componenti informative fondamentali, la tessitura e la struttura.

Sulla prima si applica, un algoritmo di texture synthesis, mentre la seconda è processata da inpainting diffusivo. Lo schema fondamentale è il seguente

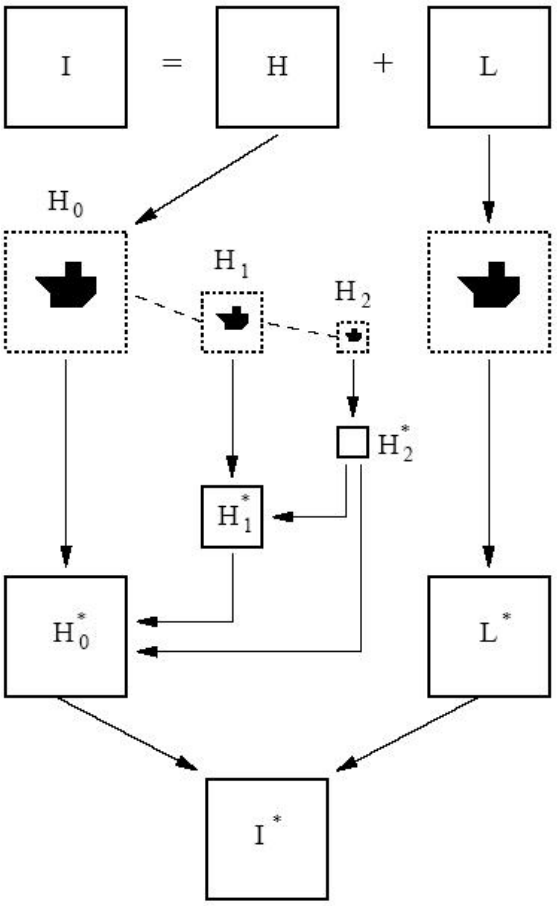


Fig. 12.2 - Schema fondamentale dell'algoritmo. Immagine scomposta. Singola parti restaurate. Ricomposizione dell'immagine originale.

Descriviamo l'algoritmo in maniera generale, focalizzandone i punti principali, che andremo a studiare in dettaglio successivamente.

1. L'immagine in ingresso I è decomposta in parte ad alto contrasto H e in una parte a basso contrasto L utilizzando la trasformata coseno discreta (DCT);

2. L'algoritmo "Fast Digital Image Inpainting" [9] è applicato all'interno dell'area coperta dalla maschera sull'immagine L, ottenendo l'immagine restaurata L^*. Durante questo passo viene utilizzata l'informazione proveniente dall'intera are di input, ma solo la parte coperta dalla maschera viene modificata;

3. L'immagine H è decomposta in una piramide Gaussiana con n+1 livelli H_i (i=0,...,n);

4. Iniziando dal più alto livello H_n applichiamo l'algoritmo di texture synthesis [10] all'interno delle aree mascherate nelle H_i. Per ogni livello si ottiene la relativa immagine restaurata H_i^*.

5. Le due immagini ottenute, L^* e H_0^*, sono sommate per ottenere I^*, la versione restaurata di I.

Estrapoliamo minuziosamente le metodologie proposte e riassunte in questi cinque punti, a cominciare proprio dalla scomposizione dell'immagine.
Si è discusso parecchio riguardo alle diverse componenti che formano l'immagine, L e H. Queste contengono differenti informazioni che rendono piacevole la vista dell'immagine se unite, differenti informazioni che costituiscono distinte componenti frequenziali.

Questo cosa significa? In una immagine, quando ci inbatte in zone ad alto contrasto, si è incappati in un'area ad elevato contenuto frequenziale. In quella parte dell'immagine le frequenze saranno di maggior valore rispetto alle zone uniformi, caratterizzate da basse frequenze.
Da questo ragionamento un'idea salta subito all'occhio: per ottenere due immagini distinte, una ad alta frequenza e una a bassa frequenza, basterà trasformare i pixel nel loro contenuto frequenziale e separare adeguatamente le componenti.
Questa operazione è eseguita in modo ottimale dalla trasformata coseno discreta o DCT (Discrete Cosine Transform) [18][19].
La DCT è una trasformata che, come la trasformata di Fourier, permette di passare dal dominio spaziale a quello frequenziale, con la differenza che mentre la trasformata di Fourier opera su numeri complessi, la DCT ottiene in uscita numeri appartenenti ai reali.
Le funzioni esponenziali complesse di Fourier sono sostituite da cosinusoidi reali, risultando più maneggevole e nettamente migliore per i nostri obiettivi.
La DCT applicata all'elaborazione di immagini è una trasformazione bidimensionale che consiste nell'applicazione di un operatore lineare, quindi invertibile, al segnale da elaborare in modo da rendere il più possibile diagonale la sua matrice di autocorrelazione e quindi concentrare l'informazione su un minor numero di coefficienti; in sostanza, si fanno comparire molti zeri.
La DCT diretta è così definita

$$F(u,v) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}} C(u)C(v) \sum_{x=0}^{N}\sum_{y=0}^{M} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2M}$$

Cerchiamo di capire.

$f(x,y)$ rappresenta il valore dei pixel nell'immagine originale alla posizione (x,y), N e M sono le dimensioni dell'immagine (rispettivamente numero di righe e numero di colonne), mentre i coefficienti $C(u)$ e $C(v)$:

$$C(u), C(v) = \frac{1}{\sqrt{2}} \qquad \text{se } u, v = 0$$

$$C(u), C(v) = 1 \qquad \text{altrimenti}$$

Da questa ultima condizione si può notare come vi siano tre possibili casi distinti:

- Sia *u* che *v* sono diversi da zero, la nostra equazione per il calcolo della DCT diviene semplicemente

$$F(0,0) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}}\sum_{x=0}^{N}\sum_{y=0}^{M} f(x,y)$$

  questo perché $C(u), C(v) = 1$ e $\cos\frac{(2x+1)\cdot 0\cdot \pi}{2N} = \cos\frac{(2y+1)\cdot 0\cdot \pi}{2M} = \cos 0 = 1$.

- Sia *u* che *v* sono uguali a zero, abbiamo

$$F(u,v) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}}\cdot\frac{1}{2}\cdot\sum_{x=0}^{N}\sum_{y=0}^{M} f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2M}$$

  poiché $C(u)C(v) = \sqrt{\frac{1}{2}}\cdot\sqrt{\frac{1}{2}} = \frac{1}{2}$

- E infine abbiamo i due sottocasi: se $u = 0$ e $v \neq 0$

$$F(0,v) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}} \cdot \sqrt{\frac{1}{2}} \cdot \sum_{x=0}^{N}\sum_{y=0}^{M} f(x,y)\cos\frac{(2y+1)v\pi}{2M}$$

perché $C(u)C(v) = 1 \cdot \sqrt{\frac{1}{2}}$ e $\cos\frac{(2x+1)u\pi}{2N} = \cos\frac{(2x+1)0\pi}{2N} = \cos 0 = 1$.

E il caso $u \neq 0$ e $v = 0$:

$$F(u,0) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}} \cdot \sqrt{\frac{1}{2}} \cdot \sum_{x=0}^{N}\sum_{y=0}^{M} f(x,y)\cos\frac{(2x+1)u\pi}{2N}$$

dato che $C(u)C(v) = \sqrt{\frac{1}{2}} \cdot 1$ e $\cos\frac{(2y+1)v\pi}{2M} = \cos\frac{(2y+1)0\pi}{2M} = \cos 0 = 1$.

La DCT è una trasformata invertibile senza perdita, permette di ricostruire perfettamente l'immagine originale.
La sua inversa è così definita:

$$f(x,y) = \sqrt{\frac{2}{N}}\sqrt{\frac{2}{M}}\sum_{u=0}^{N}\sum_{v=0}^{M} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2M}$$

e anche qui, in modo analogo alla trasformata diretta:

$$C(u), C(v) = \frac{1}{\sqrt{2}} \qquad \text{se } u, v = 0$$

$$C(u), C(v) = 1 \qquad \text{altrimenti}$$

creando altri tre casi simili ai precedenti, con la sola differenza che i termini $C(u)$ e $C(v)$ stanno all'interno delle sommatorie anziché all'esterno.

I coefficienti ottenuti con il calcolo della traformata coseno sono in numero pari a quello dei pixel e sono disposti in matrice in modo tale che il valor medio, ovvero la componente continua, sia nell'angolo in alto a sinistra, mentre i coefficienti delle componenti alternate, tutti gli altri, sono disposti in maniera crescente, come indicato dalla seguente figura

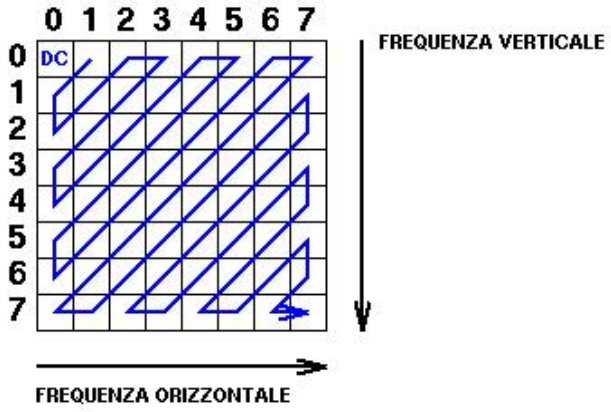


Fig.12.3

con le basse frequenze nella zona a nord-ovest della matrice e le alte frequenze situate a sud-est. Una volta computata la DCT, l'algoritmo assegna a L le prime k sottobande frequenziali. Praticamente i primi k valori in alto a sinistra nella matrice contenente i coefficienti della DCT appena calcolati, vengono copiati all'interno della matrice che conterrà i pixel alle basse frequenze.

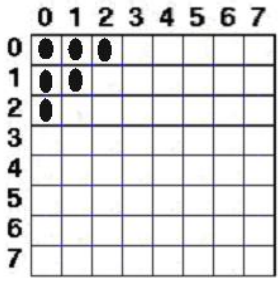


Fig. 12.4 - Vengono inseriti in L solo i valori della DCT indicati dal pallino nero, tutti gli altri sono messi a zero

Come notiamo dalla figura schematica, i coefficienti indicati col pallino nero, corrispondenti alle basse frequenze, sono quelli che vengono inseriti all'interno della matrice *L*, mentre tutti gli altri sono posti uguali a zero.
Dopodichè, per riottenere i valori dei pixel, si applica la trasformata inversa su *L*.
H è ricavata in modo semplice e immediato dalla relazione $H = I - L$, per sottrazione dell'immagine *L* dall'immagine originale.

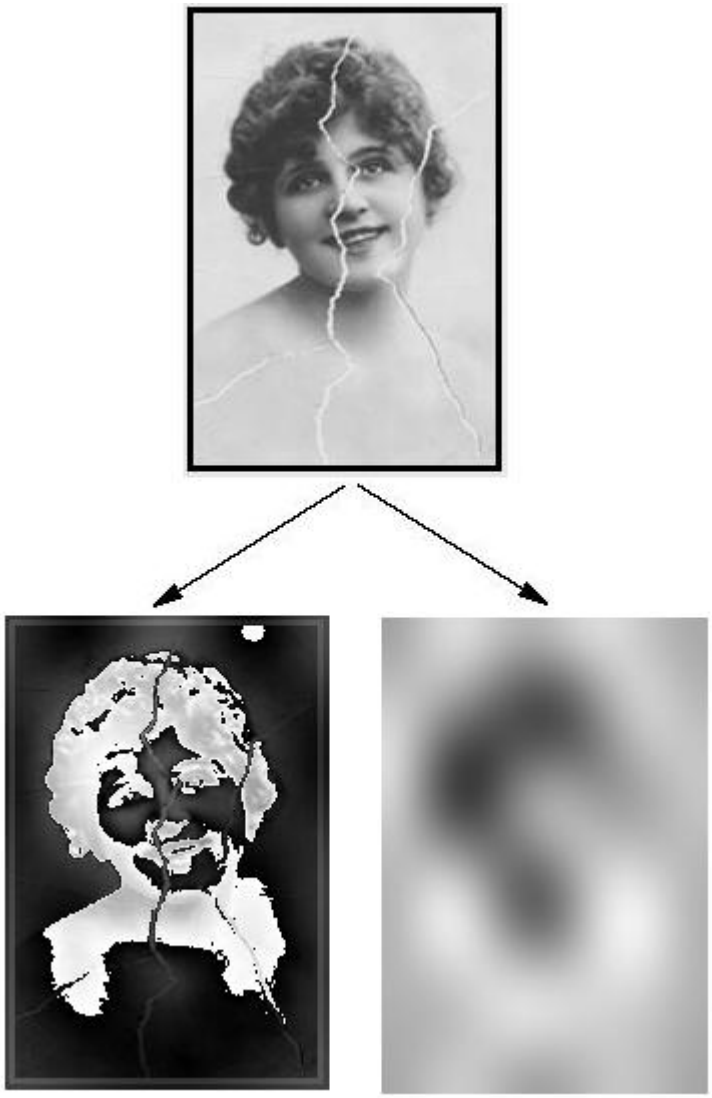


Fig. 12.5 - Risultato della scomposizione dell'immagine originale (in alto)
In basso a sinistra la componente a alta frequenza, a destra in basso la componente a bassa frequenza

Nella figura 12.5 si nota la scomposizione dell'mmagine tramite DCT.
La parte a bassa frequenza contiene per lo più i colori e può essere correttamente processata da algoritmi che ne mantengano il gradiente, come quelli a diffusione esaminati nel capitolo relativo.
L'immagine in basso a sinistra, contenente i contorni. Sarà processata dall'algoritmo di sintesi della tessitura tramite multirisoluzione [10].
I coefficienti della DCT sono quasi tutti localizzati alle basse frequenze, la componente frequenziale ad alti valori infatti è scarsa sull'intera immagine.
Si pensi a qualsiasi tipo di immagine, i contorni e le zone ad alta frequenza in generale sono distribuiti in proporzione nettamente inferiore rispetto alle aree uniformi. Consideriamo l'immagine di dimensioni 8x8 così composta:

11	16	21	25	27	27	27	27
16	23	25	28	31	28	28	28
22	27	32	25	30	28	28	28
31	33	34	32	32	31	31	31
31	33	34	32	32	31	31	31
33	33	33	33	32	29	29	29
34	34	33	35	34	29	29	29
34	34	33	33	35	30	30	30

Fig. 12.6 - Valori di colore di un'immagine 8x8

i pixel hanno valori di colore praticamente identici, l'immagine sarà quindi piuttosto uniforme e ricca di componenti a basse frequenze, come mostrato graficamente di seguito

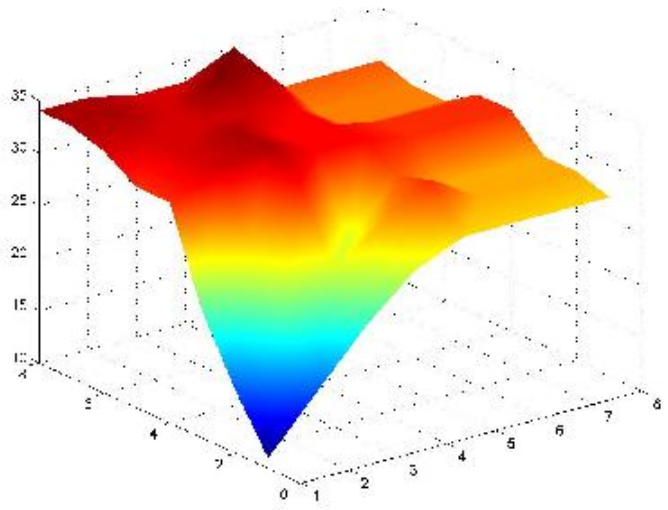


Fig. 12.7 - La stessa immagine del blocco di pixel 8x8 precedente, vista tramite un grafico tridimensionale

Se ora analizziamo il grafico della sua trasformata coseno

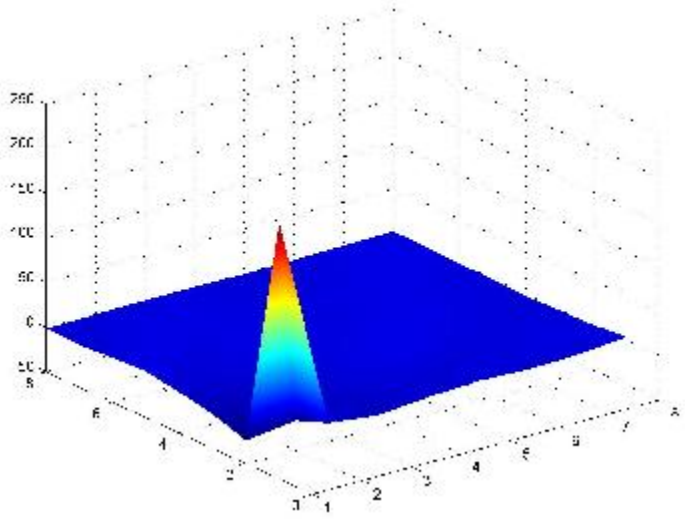


Fig. 12.8 - DCT del blocco di pixel 8x8 precedente vista tramite un grafico tridimensionale

Notiamo che valori significativi sono localizzati alle prime posizioni della matrice, mentre i restanti coefficienti sono prossimi a zero.
Bisogna per ciò fare attenzione al numero di sottobande che si assegnano all'immagine *L* durante la scomposizione, in modo da non perdere dettagli importanti che ne renderebbero distorta la ricostruzione.

Il calcolo della trasformata coseno su immagini grandi è un'operazione ad elevata complessità temporale. Per ogni pixel è necessario visitare tutti i pixel dell'immagine al fine di ottenere il valore corretto della DCT per quel singolo pixel, come è evidente dalla formula. Quindi, per immagini di dimensioni generiche MxN=Q, per ognuno dei Q pixel dovrò eseguire Q iterazioni. Si pensi per semplicità ad una piccolissima immagine di dimensioni 8x8=64, per calcolare ognuno dei 64 valori della DCT in uscita sarà necessario visitare tutti i 64 pixel, per un totale di 4086 iterazioni. Questo valore cresce spaventosamente con il crescere delle dimensioni dell'immagine. Inoltre il tempo raddoppia poiche bisogna includere anche il calcolo della DCT inversa, per cui valgono le stesse identiche considerazioni della trasformata diretta.
Un modo per risparmiare in velocità potrebbe essere quello di inserire una *look-up table* contenente i valori relativi ai due coseni nella formula della DCT, ma l'argomento del coseno dipende dalle dimensioni dell'immagine, che non è nota a priori, necessitando di un'allocazione dinamica della LUT. Il guadagno temporale è limitato.
La DCT è utilizzata come elemento fondamentale in algoritmi di compressione di immagini come JPEG (Joint Picture Expert Group) o video come MPEG (Moving Picture Expert Group). In questo tipo di algoritmi si adopera una sorta di "trucco" per velocizzare il computo della trasformata coseno: invece di calcolarla sull'intera immagine questa viene divisa in blocchi di dimensioni 8x8, su ognuno di questi blocchi si calcola la DCT separatamente e in modo autonomo.

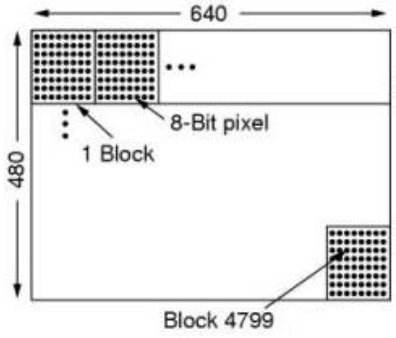


Fig. 12.9 - Divisione dell'immagine in blocchi di dimensioni 8x8

Su ogni blocco si procede con lo stesso meccanismo di calcolo della DCT, dividendo le basse frequenze dalle alte, come se ogni blocco fosse un'immagine a sè stante.
Così facendo il risultato ottenuto è strabiliantemente veloce e si ha, ovviamente, ancora scomposizione dell'immagine nelle sue componenti frequenziali, come possiamo vedere:

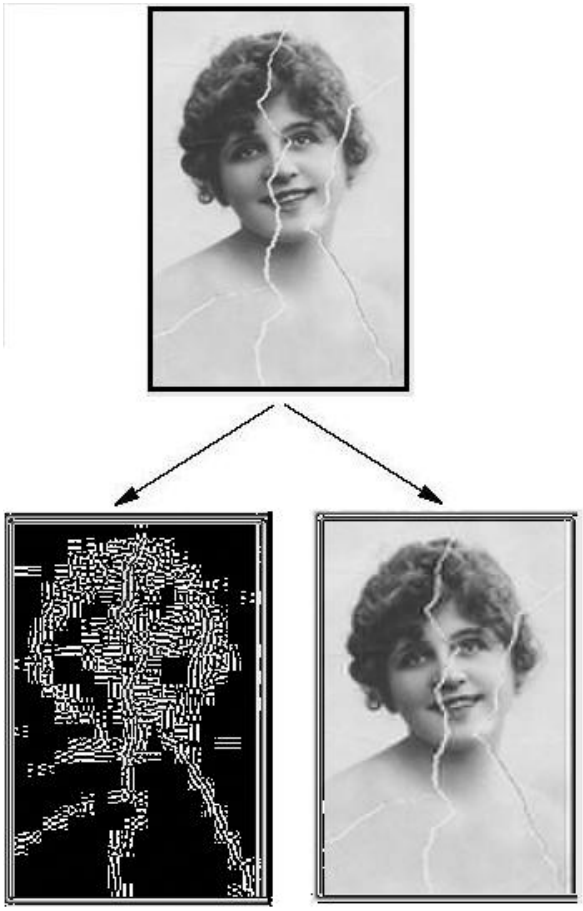


Fig. 12.10 - Scomposizione dell'immagine nelle sue componenti frequenziali con la DCT su blocchi di 8x8 pixel

Purtroppo questa scomposizione non fa al caso nostro.
Gli autori stessi dell'algoritmo [17] richiedono che la DCT sia calcolata sull'intera immagine. Effettivamente, proponendo in ingresso all'algoritmo le immagini contenute nella figura precedente, il risultato ottenuto è pessimo e sono ben

evidenti artefatti sgradevoli che non appaiono se l'immagine è scomposta calcolando la trasformata coseno lungo tutta l'immagine.
Una volta che l'immagine è stata scomposta si applicano tecniche differenti su L e su H.
La parte composta dalle basse frequenze è sottoposta all'algoritmo analizzato nei capitoli precedenti, Fast Digital Image Inpainting, un algoritmo che permette di mantenere i gradienti di colore in modo efficiente e che opera velocemente e con buoni risultati qualora non vi siano aree ricche di contrasti.
Dall'immagine H viene creata una piramide Gaussiana con al più tre livelli, ognuno dei livelli ottenuto è sottoposto all'algoritmo di sintesi della tessitura, come proposto in [10], applicando la tecnica della multirisoluzione. In questo modo solo al livello più alto (quello con minor risoluzione) si applica lo schema di texture synthesis a singola risoluzione, su tutti gli altri livelli si opera tenendo conto dei risultati forniti ai livelli precedenti, come indicato nel capitolo 10.
Terminata l'esecuzione dei due algoritmi descritti sulle rispettive immagini, L e H, e ottenute le immagini adeguatamente filtrate in uscita, L^* e H^*, la ricostruzione dell'immagine originale filtrata I^* è data da una semplice somma delle sue due singole componenti restaurate: $I^* = L^* + H^*$.
L'algoritmo qui proposto permette di ottenere risultati molto buoni su un'ampia varietà di immagini, anche se a volte si ottengo degli sfocamenti dovuti alla diffusione, in modo particolare se non si è divisa correttamente l'immagine, lasciando elementi della tessitura all'interno di L, che va così persa nella maggior parte dei casi.
La pecca più grande di questo algoritmo risiede, come gia accennato, nella sua lentezza d'esecuzione, dovuta al calcolo di DCT e DCT inversa.
Pur sfruttando due algoritmi di restauro tra i più veloci, [9] e [10], come il nome stesso evidenzia, che operano nell'ordine di pochissimi secondi (molte volte meno di un secondo), l'algoritmo di Yamauchi et al. difficilmente scende sotto il tempo dei due minuti d'esecuzione, anche per immagini di piccole dimensioni.

13. Alcuni risultati ottenuti

Mostreremo ora una carrellata di risultati ottenuti con l'applicazione degli algoritmi descritti, visualizzando casistiche in cui l'algoritmo ha operato in modo ottimale e altre di fallimento. Inoltre si descriveranno i relativi tempi di esecuzione. Iniziamo con il primo metodo proposto, [7] :

Fig. 12.11 - Immagine orginale affetta da crepa lungo la sua superficie
Si può notare come la crepa attraversi sia zone ad alta frequenza che altre a bassa frequaenze

Fig. 12.12 - Immagine restaurata con l'algoritmo proposto in un tempo di circa 5 minuti
(le dimensioni delle immagini sono 525 x 441)

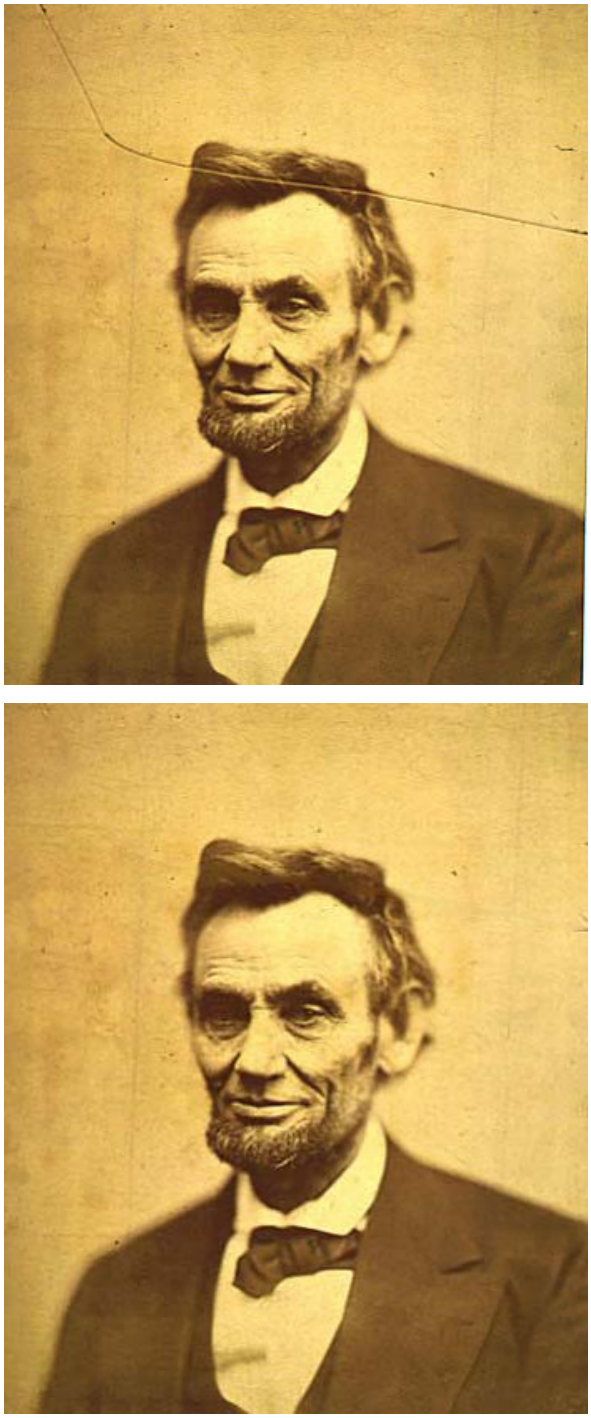


Fig. 12.13 - Sopra l"mmagine originale con crepa. Sotto l'immagine restaurata con l'algoritmo Image Inpainting
Il tempo di esecuzione è stato di circa un minuto (le dimensioni delle immagini sono 350 x 426)

Fig. 12.14 - Immagine originale con crepa

Fig. 12.15 - Immagine filtrata con l'algoritmo di Bertalmio in circa 6 minuti (le dimensioni delle immagini sono 597 x 577)

Si può notare come i risultati siano buoni ma con tempi di esecusione estremamente lunghi.
Mostreremo ora risultati molto simili, sulle stesse immagini, ottenuti con l'algoritmo Fast Digital Image Inpainting [9] in tempi nettamente inferiori.
Certamente può succedere che i risultati siano diversi tra l'output di [7] e quello di [8], inoltre a volte risulta complesso il meccanismo di creazione della barriere, ma se si paragonano i tempi di esecuzione, l'algoritmo di Oliveira et al. è certamente vincente su quello di Bertalmio et al.

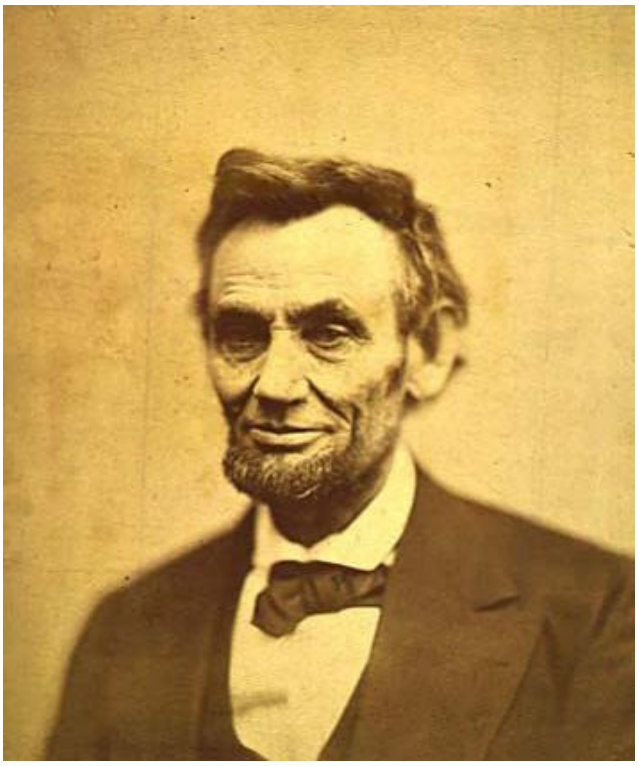
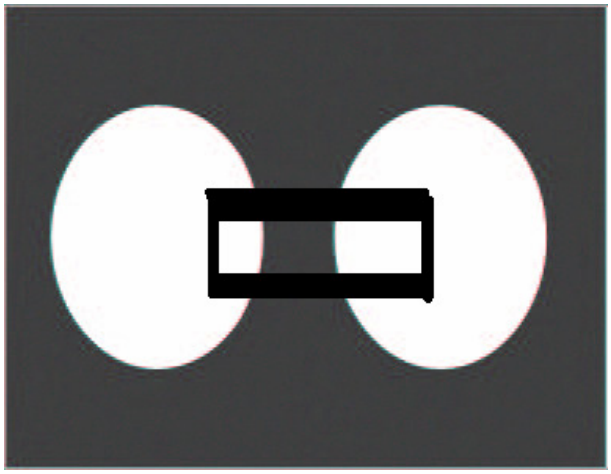


Fig. 12.16 - Immagine di Lincoln restaurata con FDII

Fig. 12.17 - Si può notare come queste due immagini diano risultati quasi identici a quelli ottenuti con [7] ma riducendo i tempi. Le due immagini sono state infatti ottenute in 0.1 secondi e 1.5 secondi rispettivamente

Gli algoritmi di diffusione proposti falliscono quando l'area di inpainting aumenta di grandezza



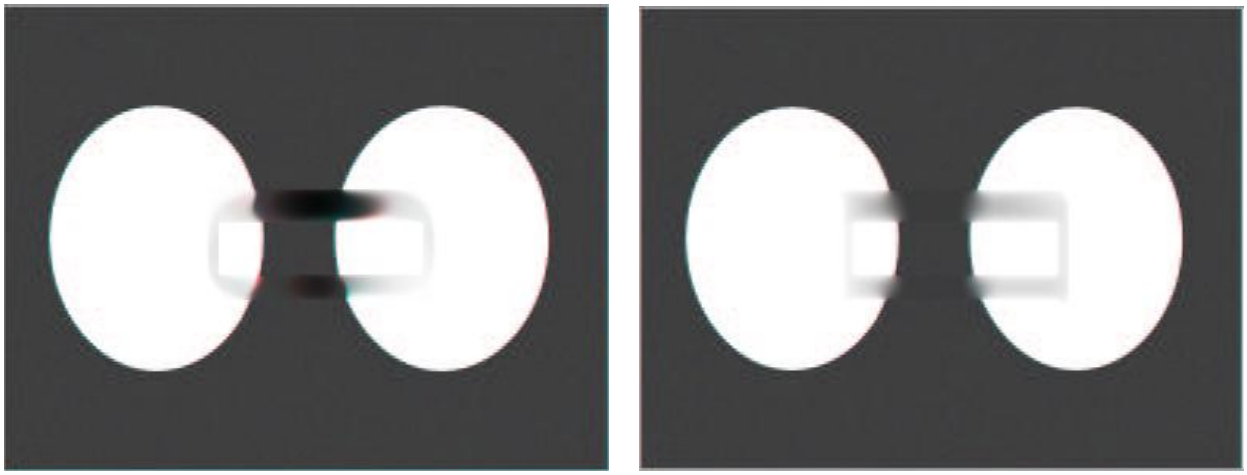


Fig. 12.18 - Immagine originale in alto, in basso a sinistra immagine filtrata con [7], in basso a destra si è usato [9] (le dimensioni delle immagini sono 383 x 295)

Vediamo cosa succede se la stessa immagine viene filtrata con l'algoritmo di Wei e Levoy proposto in [10]

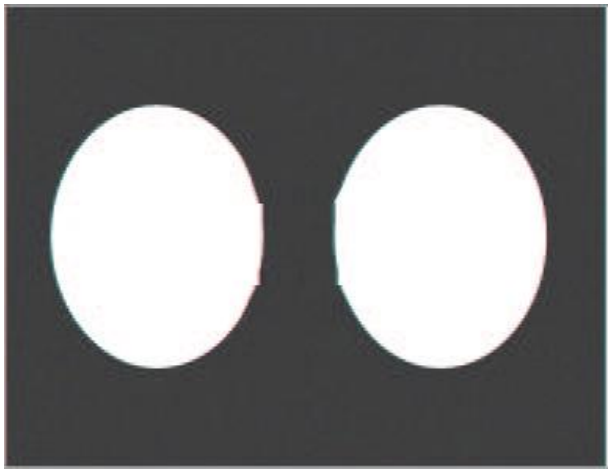


Fig. 12.19 - La stessa immagine di prima computata dall'algoritmo [10]

Questo algoritmo oltre ad essere estremamente veloce è anche molto efficiente, come dimostrano i seguenti esempi

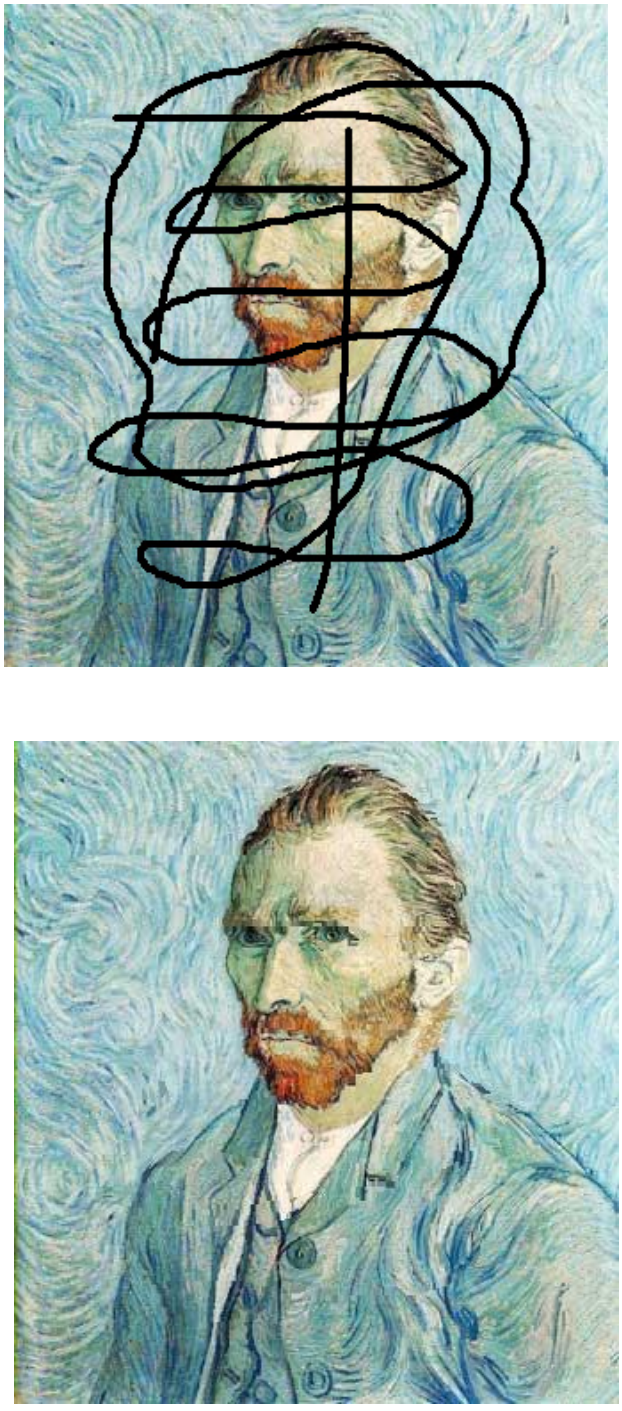


Fig.12.20 (le dimensioni delle immagini sono 340 x 372)

Fig.12.21 (le dimensioni delle immagini sono 512 x 380)

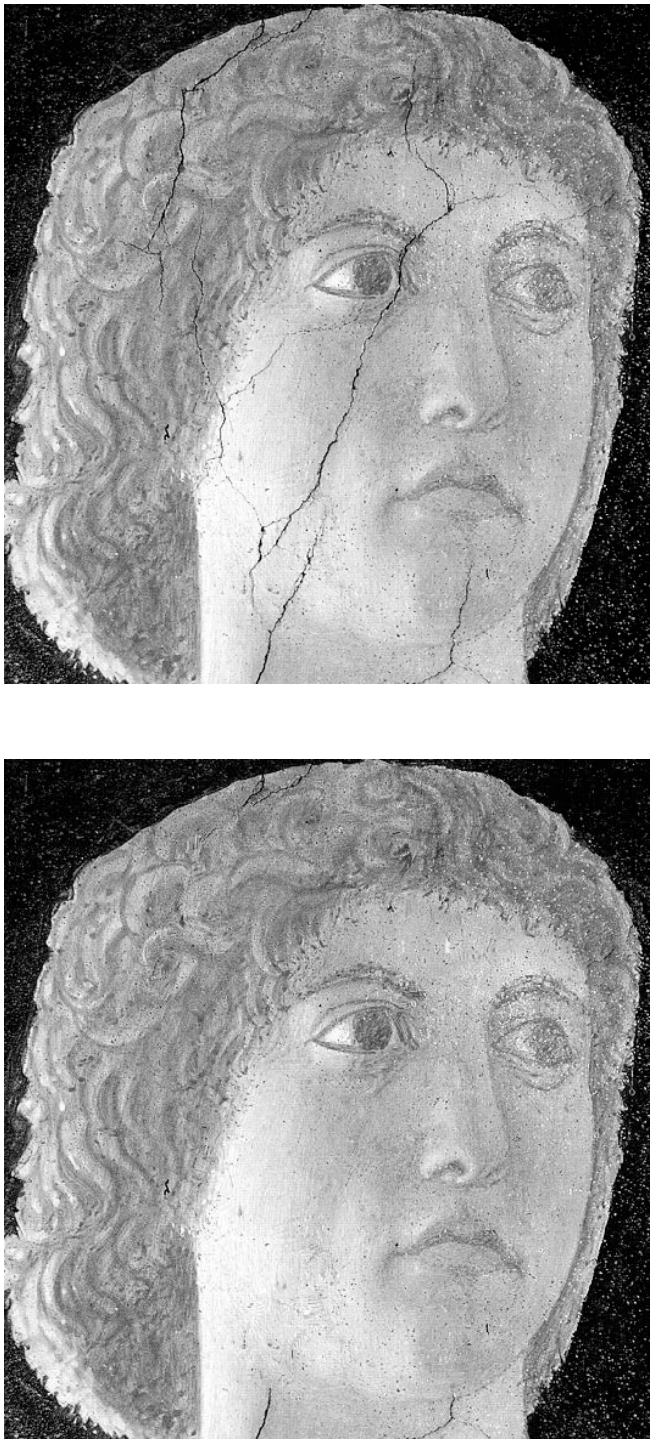


Fig.12.22 (le dimensioni delle immagini sono 530 x 560)

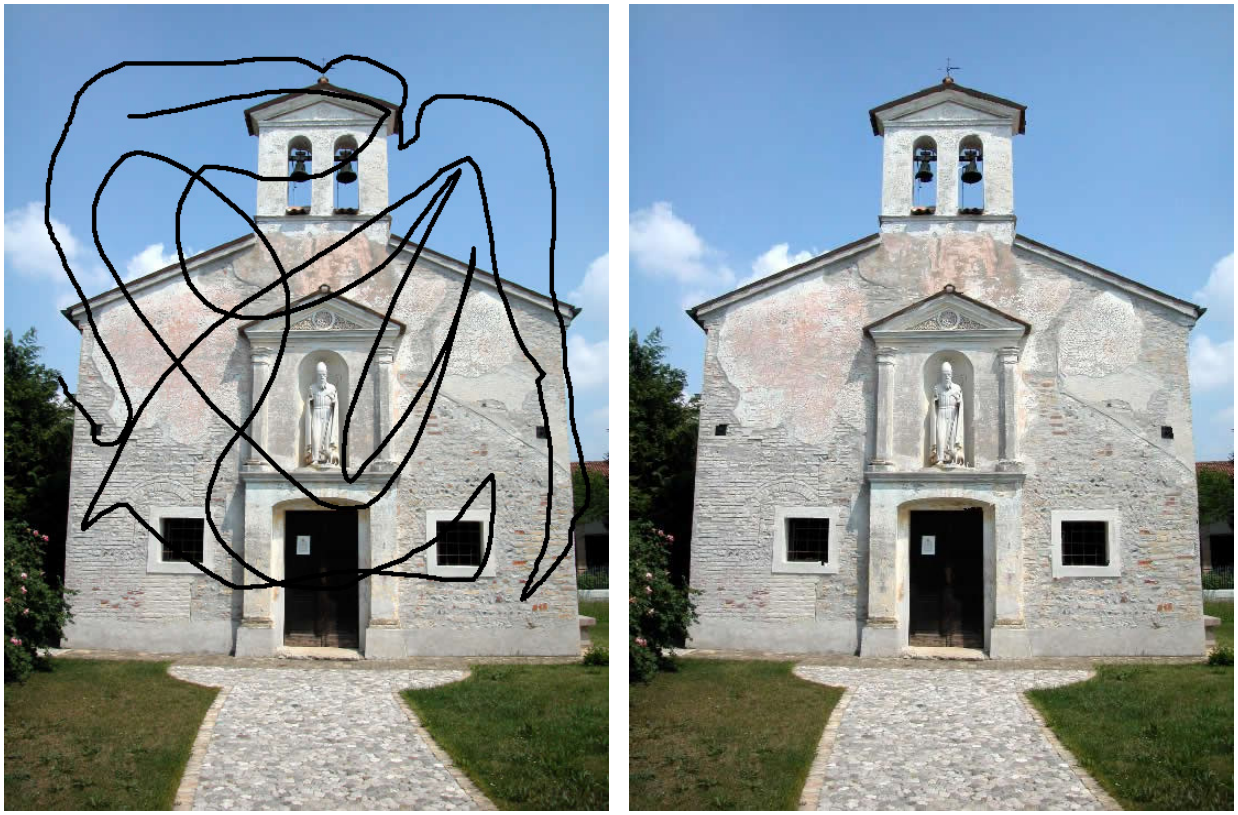


Fig.12.23 (le dimensioni delle immagini sono 550 x 733)

Abbiamo mostrato una carrellata di immagini difettose, spesso appositamente alterate per prova, e ne abbiamo fornito un loro restauro tramite l'algoritmo "Fast Texture Synthesis using Tree-Structured Vector Quantization".
I risultati sono davvero buoni e i tempi di esecuzioni davvero ristretti, si va da un massimo di quattro secondi per l'immagine della chiesa ad un minimo di 0.5 secondi per l'immagine con la scritta "Hello World Bye".

Vediamo ora un caso in cui questo algoritmo fallisce

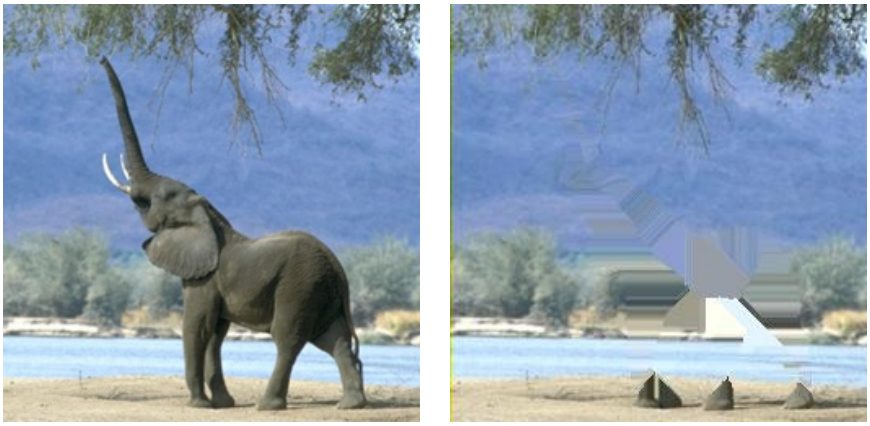


Fig. 12.24 - Immagine originale e immagine filtrata. Si nota la creazione di artefatti (le dimensioni delle immagini sono 256 x 256)

Tentando di eliminare l'elefante dall'immagine, lasciando solo lo sfondo, so ottengono dei pessimi artefatti, specialmente nella zona superiore al fiume. Una soluzione si può ottenere semplicemente copiando intere "pezze" di pixel anziché un singolo pixel per volte, come spiegato nel capitolo sulla priorità, parlando dell'algoritmo di Criminisi et al.
Il risultato ottenuto è il seguente

Fig.12.25

Esponendo questo algoritmo basato su priorià si disse che il restauro iniziava proprio dalle aree ad alte frequenze, come linee e contorni, che avessero il maggior numero di vicini disponibili, come si può notare di seguito

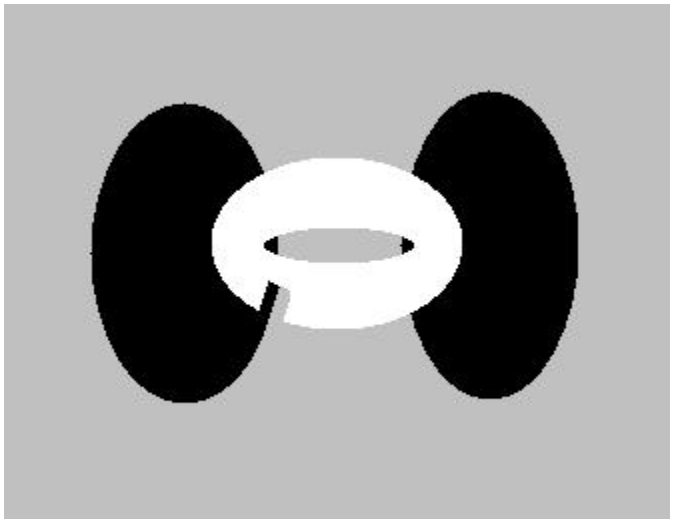


Fig. 12.26 - In questo caso l'algoritmo è stato fermato dopo alcune iterazioni, in modo da rendere evidente la tecnica di priorità che predilige il polungamento delle linee ad alto contrasto che incontrano il bordo della maschera all'interno del difetto

Lasciando eseguire l'algoritmo fino al suo termine si ottiene l'immagine seguente, perfettamente restaurata

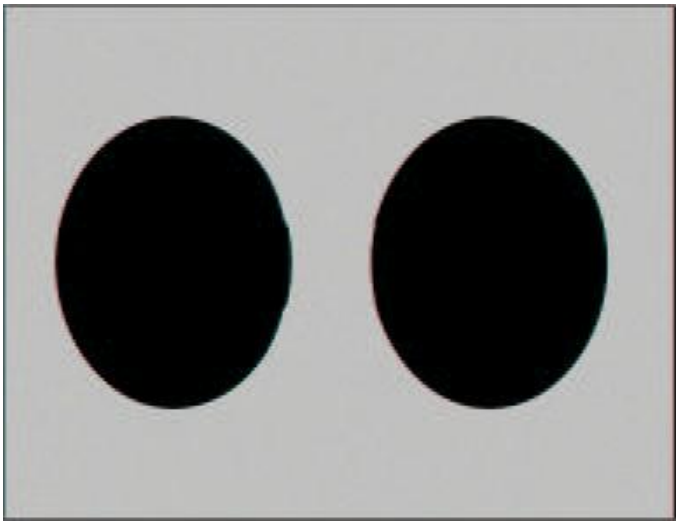


Fig. 12.27 - Immagine precedente modificata con il metodo proposto in [12]

Ecco ora alcuni esempi di restauro di crepe tramite l'utilizzo dell'algoritmo di Marco Ortolan [15]

Fig. 12.28 - Immagine difettosa e immagine perfettamente restaurata (le dimensioni delle immagini sono 386 x 343)

Ricordiamo che l'algoritmo di Ortolan è stato studiato e realizato appositamente per il restauro di crepe su vecchie stampe fotografiche.
Un altro esempio

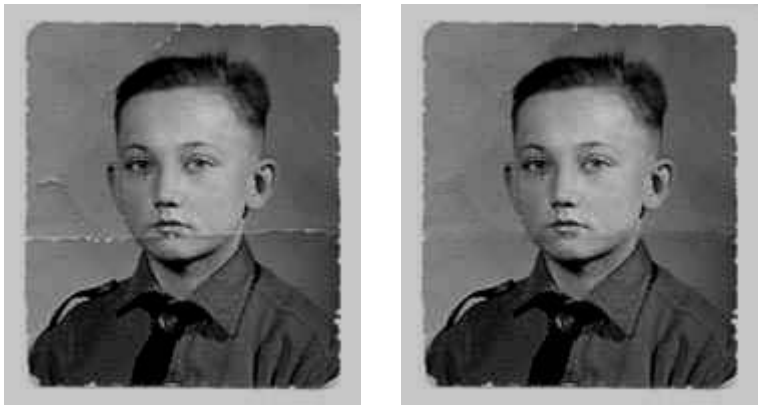


Fig.12.29 (le dimensioni delle immagini sono 178 x 200)

E un altro

Fig.12.30 (le dimensioni delle immagini sono 328 x 327)

Ora un caso in cui invece questo algoritmo non risulta essere ottimale per via della sua componente diffusiva.

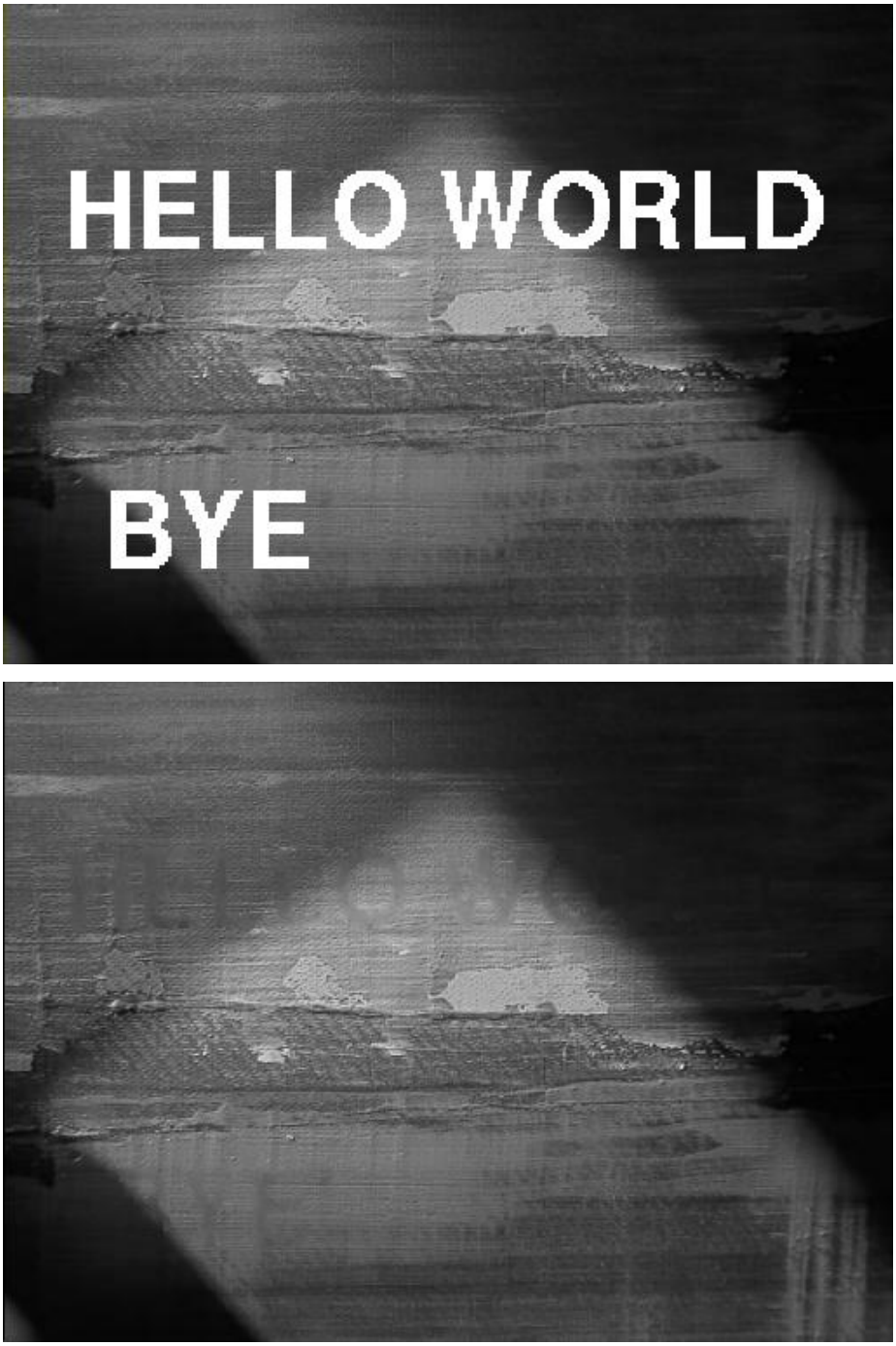


Fig. 12.31 - Si nota ancora la scritta “Hello World Bye” sullo sfondo

14. Conclusioni e lavori futuri

Sono stati illustrate le classi principali di algoritmo per il restauro digitale attualmente in circolazione. Questi algoritmi sono stati implementati e ne sono stati evidenziati e analizzati alcuni casi particolari.
Tutti gli algoritmi sono stati implementati in C++, utilizzando Visual Studio 2005.
La scelta del linguaggio C è dovuta alla sua semplicità e alla velocità ottenuta in fase di test. Un'altra scelta plausibile sarebbe potuta essere quella di utilizzare Matlab. Questo permette di semplificare notevolmente la scrittura degli algoritmi, fornendo un'interfaccia altamente pratica all'elaborazione delle immagini, ma incrementando a dismisura i tempi di esecuzione.
Se mi si chiedesse di scegliere tra tutti quelli esaminati un "metodo preferito", risponderei senza esitare: l'algoritmo di Wei e Levoy, "Fast Texture Synthesis using Tree-Structured Vector Quantization".
Le sue doti di velocità, unite agli ottimi risultati ottenuti, oltre alla semplicità computazionale e implementativa, ne fanno un metodo rapido ed intuitivo, in grado di fornire risultati sorpendenti nel campo del restauro di immagini.
I risultati non sono altrettanto buoni se si passa all'ambito degli effetti speciali, in cui è spesso richiesta l'eliminazione di soggetti da un'inquadratura, specialmente quando i soggetti sono di grosse dimensioni, come mostrato nel capitolo precedente. Ma questo fuorvia dall'obiettivo della tesi.
L'algoritmo [10] è nettamente estrapolato dal suo ambito originale, la sintesi della tessitura e ben applicato al restauro di immagini.
Viene naturale pensare che non sia un caso a sé, ma che tutti gli algoritmi di texture synthesis possano essere applicati all'inpainitng. La questione potrebbe rappresentare un nuovo caso di studio. Un ottimo punto di partenza è dato dall'algoritmo di Michael Ashikhmin [20] che pone le sue basi proprio

nell'algoritmo [10], modificandolo in maniera da risultare migliore sulla sintesi di tessiture appartenenti ad immagini naturali, non create artificialmente ma prese dal mondo reale, e quindi perfettamente adattabile al mondo della fotografia.
Un altro interessante spunto si ha in un altro algoritmo di Bertalmio: "Simultaneous Structure and Texture Image Inpainting", [21], come il procedimento di Yamauchi et al. discusso in questa tesi, si propone di suddividere l'immagine nelle sue diverse componenti frequenziali e di applicare diffusione alle basse frequenze e texture synthesis alle alte frequenze.
Le differenze rispetto a [17] risiedono nel diverso procedimento matematico utilizzato per la scomposizione dell'immagine originale, nell'algoritmo di diffusione utilizzato (adopera l'altro algoritmo di Bertalmio qui descritto [7]) e nella diversa procedura di sintesi della tessitura eseguita sulle alte frequenze.
Da quanto inteso dall'algoritmo [17] e dalla lettura dell'articolo in [21], una volta scomposta l'immagine, si possono applicare tutti gli algoritmi di diffusione e texture synthesis conosciuti, fino a trovare la combinazione migliore.
Campi di studio potrebbero comprendere una più veloce divisione dell'immagine e la ricerca di un perfetto equilibro tra le immagini ottenute dalla scomposizione.
Ancora, l'algoritmo di Ortolan [15] è stato studiato appositamente per la rimozione di crepe su vecchie stampe digitali, come più volte ribadito.
Nel corso della tesi in cui è stato proposto, non si è assolutamente accennato ad una eventuale implementazione su immagini a colori. Quando si è cercato di realizzarlo nacque un problema dovuto al calcolo della priorità tramite varianza. Solitamente, trattando immagini a colori, si opera per livelli, sottoponendo le diverse componenti dell'immagine all'algoritmo. Ricordando che la priorità data dalla varianza va a cercare il pixel che possiede la varianza massima nel suo intorno.
Ci si è così trovati di fronte ad un problema: su talune immagini, per una componente la varianza massima corrispondeva ad un certo pixel, mentre per altre componenti i pixel prioritari erano localizzati diversamente. Lasciando

operare così l'algoritmo, si avrebbero avute modifiche su pixel differenti nei diversi livelli, rischiando di sfalsare i colori una volta ricomposta l'immagine iniziale.
Trovare una soluzione a questo problema potrebbe essere un altro appiglio per un caso di studio.
Una semplice proposta balzata alla mente potrebbe consistere nel scegliere, in modo del tutto casuale, di tenere conto della priorità ottenuta su uno solo dei tre livelli e operare modifiche al punto sito in quella posizione anche sugli altri due livelli. Un'ultima poroposta riguarda le maschere.
Il restauro digitale è strettamente legato all'utilizzo di maschere per l'evidenziazione del difetto all'interno dell'immagine, data la complessa natura degli stessi che ne rende alquanto difficile il riconoscimento automatico.
Realizzare maschere in modo "manuale" diviene un meccanismo alquanto impreciso, per via della natura umana dell'operazione.
Molti algoritmi sono fortemente influenzati dalla precisione nella creazione delle maschere: se troppo piccole si rischia di tralasciare alcuni pixel danneggiati, creando effetti indesiderati, se troppo grandi si rischia di togliere informazioni utili al restauro.
Marco Ortolan, nella sua tesi, propone un meccanismo di *riduzione della maschera* molto semplice: analizza tutti i pixel coperti dal bordo della maschera e verifica quali punti sono riattribuibili all'immagine e quali invece sono da considerarsi di diritto facenti parte della maschera.
Per fare questo si devono analizzare i punti integri adiacenti a quello in esame estrapolandone l'informazione di massima, utilizzando la loro media, per verificare se il pixel da analizzare sia a loro simile oppure no. Nel caso di somiglianza si può supporre che il pixel sia stato compreso nella zona da ricostruire erroneamente, mentre nel caso in cui il pixel si discosti molto da tale valore medio si può supporre che il punto appartenga di diritto al difetto e quindi debba essere mantenuto coperto dalla maschera.

Il fatto di considerare solo i punti confinanti risiede nel fatto che si vuole fare una ricerca di tipo puntuale e considerare zone di rierca più vaste rischierebbe di comprendere dati che fuorvierebbero dal risultato voluto.
Una matrice tre per tre viene fatta scorrere su tutto il bordo della maschera, disposta di volta in volta in modo tale che il riquadro centrale sia posizionato sul punto indagato. La matrice viene riempita con i valori dei punti dell'immagine non coperti dalla maschera; di seguito si calcola la media di tali valori così da confrontarla con il punto centrale dell'immagine coperto dal bordo della maschera secondo la seguente modalità: se si verifica che

$$| p(x,y) - m(x,y) | \leq \alpha$$

allora la maschera viene cancellata nel punto esaminato, altrimenti essa viene lasciata immutata.
Con p(x,y) si intende il valore del punto considerato di coordinate x e y, m(x,y) è la media dei punti appartenenti alla matrice centrata nel punto p non coperti dalla maschera e a è il coefficiente di soglia, cioè una costante intera dipendente dalle caratteristiche dell'immagine e da quelle del difetto (più il danno si avvicina alle caratteristiche dell'immagine e più piccola sarà la costante in quanto la soglia di intervento si fa più stringente).
Questa è una semplice tecniche per la riduzione della maschera che da buoni risultati.
Per poterla applicare ovviamente la maschera deve essere più ampia del difetto.
Sulla scia di questa si potrebbe provare a studiare e implementare nuove metodologie per una ricerca migliore del difetto data una maschera disegnata manualmente dall'utente.

Appendice

Questo appendice è creato per facilitire la comprensione di alcuni concetti contenuti nella tesi.
Inoltre ritengo possa essere uno strumento utile a chi si avvicina per la prima all'elaborazione di immagini, sia da un lato teorico che da un lato applicativo e implementativo.
Le immagini bitmap, o immagini raster, sono definite da una griglia (mappa di bit) di piccoli quadratini detti pixel (Picture Element). A ciascun pixel corrispondono un valore cromatico e due coordinate spaziali.
Sono le immagini che meglio riproducono le sfumature dettagliate caratteristiche di foto e dipinti, ma possono perdere dettaglio e peggiorare in qualità se ingrandite o stampate con scarsa risoluzione. Le dimensioni assolute di un'immagine bitmap dipendono quindi direttamente dal numero di pixel che la compongono lungo l'altezza e la larghezza. Un fattore cruciale per le immagini bitmap è la risoluzione ossia il numero di pixel per unità di lunghezza che le definiscono. La risoluzione si esprime in pixel x pollice (ppi) o in punti x pollice (dpi). Dimensione e risoluzione sono quantità legate da proporzionalità inversa; un'immagine ad alta risoluzione contiene pixel più piccoli e in maggior numero rispetto ad un'immagine delle stesse dimensioni ma con risoluzione inferiore. Appare ovvio quindi che maggiore è la risoluzione, migliore è la qualità dell'immagine (e maggiore il suo peso); un elevato numero di pixel per unità di superficie permette di riprodurre maggiori dettagli e sfumature di colore migliori. Tuttavia, aumentare per via informatica la risoluzione ad un'originale di bassa qualità (sia esso acquisito a scanner o proveniente da una raccolta di immagini), non ne migliora le caratteristiche poichè le stesse informazioni verrebbero semplicemente distribuite tra un numero maggiore di pixel, senza aggiungere ulteriore dettaglio. Nel decidere la risoluzione migliore con cui salvare

un'immagine digitale, bisogna considerare il dispositivo di output al quale è destinata. Se essa è destinata alla visualizzazione su monitor (internet, prodotto multimediale) è sufficiente che la sua risoluzione corrisponda a quella tipica dei monitor: 72 o 96 dpi. Per risoluzione di un monitor si intende il numero di pixel (punti) per pollice lineare presenti sullo schermo, 72 dpi per Macintosh, 96 dpi per PC appunto, ma si intende anche il numero di pixel visualizzati sullo schermo orizzontalmente e verticalmente nello stesso momento.

La risoluzione dipende quindi dalle dimensioni dello schermo e dalle sue impostazioni in pixel, per esempio un comune monitor da 13 pollici è definito da 640 pixel in orizzontale x 480 pixel in verticale.

Dimensioni e risoluzione dell'immagine e risoluzione del monitor sono variabili da cui dipende la modalità di visualizzazione a schermo di un files bitmap. Un'immagine con dimensione 640x480 pixel a 72 punti di risoluzione riempie completamente un video 640x480, la stessa immagine riempirebbe comunque un monitor più grande con le stesse impostazioni in pixel di 640x480, ma i pixel apparirebbero più grandi; un'immagine con dimensione 640x480 pixel a 144 punti di risoluzione occuperebbe uno spazio quattro volte più grande (doppio sia in altezza che in larghezza) su un video 640x480.

Se l'immagine è destinata alla stampa (stampante o fotounità) la risoluzione ottimale da utilizzare dipende dalle caratteristiche della periferica di output. Per risoluzione, in questo caso, si intende il numero di punti per pollice (dpi) che il dispositivo è in grado di produrre per descrivere l'immagine.

La maggior parte delle stampanti ha risoluzioni variabili da 300 dpi a 600 dpi, le fotounità da 1200 dpi a 3200 dpi. Documenti stampati con risoluzione troppo bassa rispetto alla periferica presenteranno pixel grandi e grossolani, documenti a risoluzione troppo alta impiegheranno molto tempo per essere processati ma la qualita non potra superare quella garantita dalle specifiche hardware. Le immagini solitamente sono di due tipi, in bianco e nero e a colori.

I valori assunti dai pixel delle immagini in bianco e nero sono copresi tra 0 (nero) e 255 (bianco), corrispondenti ai livelli di grigio.
La immagini a colori sono solitamente memorizzate utilizzando 24 bit per ogni pixel (8 bit per quelle in bianco e nero). Ogni immagine a colori può essere scomposta nei suoi livelli. Se si utilizza la sintesi additiva del colore tramite lo spazio RGB (Red Green Blue) ognuno dei tre colori utilizzato conterrà la componente di colore relativa. Sommando insieme le tre componenti si ottiene il valore di colore reale che il pixel possiede.

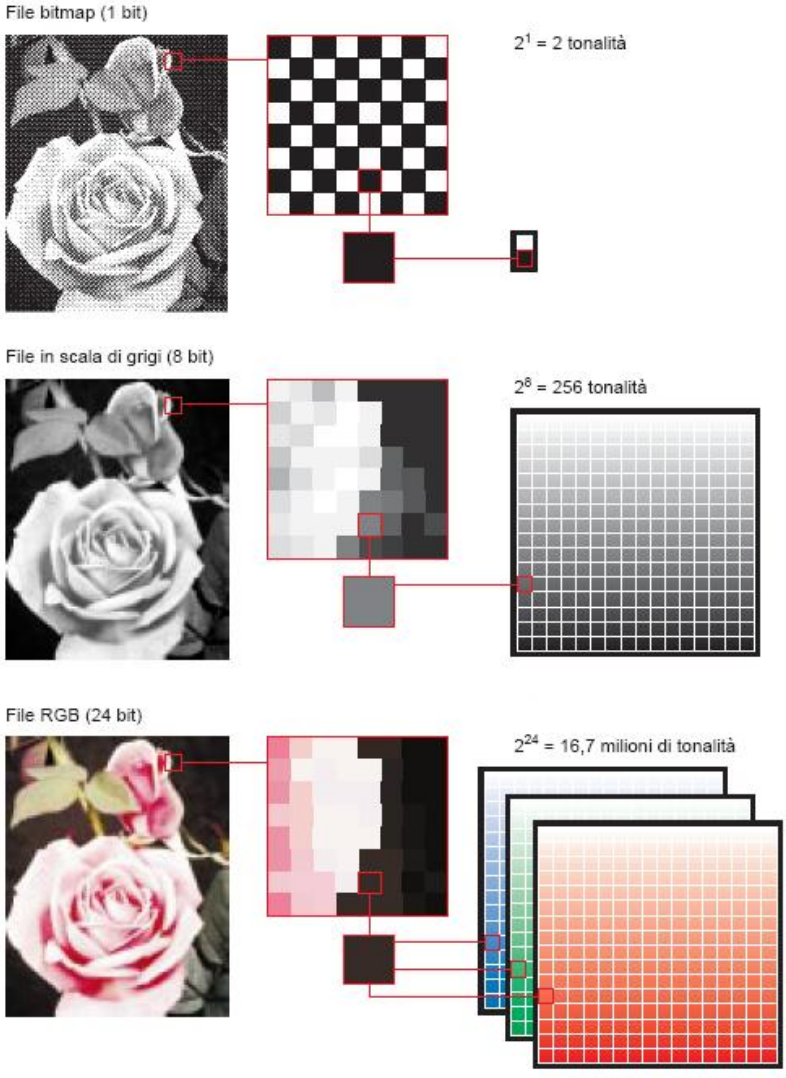


Fig. A.1 - La figura mostra tre differenti rappresenzationi dei pixel di uniimmagine, monocromatica, scala di grigi, a colori.

Scomponendo un'immagine RGB come la seguente

Fig. A.2

nelle sue componenti di colore avremmo

Fig. A.3

che rappresentano rispettivamente la componente rossa, la verde e la blu.
Da quanto sin qui affermato si intuisce perchè il metodo più utilizzato per maneggiare le immagini ed elaborarle sia tramite la forma matriciale.
Nella versione 2005 di Visual Studio è presente la libreria "atlimage.h" che permette di lavorare sulle immagini utilizzando la comodissima classe Cimage.

Questa classe fornisce degli strumenti utilissimi e molto semplici per la manipolazione delle immagini.
Innanzitutto vanno definite due variabili

```
CImage                  Image;
unsigned char           **pImage;
```

la variabile *unsigned char* reppresenta la nostra matrice. Conterrà i valori di grigio dei pixel dell'immagine. Il tipo *unsigned char* risulta essere l'ideale per questo tipo di dati, poiché il suo range di valore è proprio 0-255 (mentre il char con segno va -128 a 127).
La CImage conterrà invece l'immagine vera e propria.
Aprire e salvare l'immagine diviene facilissimo grazie a questa classe, basta usare le semplici istruzioni

```
Image.Load(NomeFile);
Image.Save(NomeFile);
```

La prima carica in Image l'immagine "NomeFile", mentre la seconda salva l'immagine col nome di "NomeFile".
Per poter lavorare sui pixel bisogna ora copiare il contenuto dell'immagine all'interno della matrice.
Anche questa è una semplice azione eseguibile con le istruzioni:

```
for (nRow=0; nRow< Image.GetHeight(); nRow++)
   for (nCol=0; nCol< Image.GetWidth(); nCol++)
   {
         pChar=(unsigned char *)(this->Image.GetPixelAddress(nCol,nRow));

         pImage[nRow][nCol]=*pChar;
   }
```

I due cicli operano in modo da visitare ogni pixel dell'immagine, muovendosi per righe e colonne, per tutta la sua larghezza (`Image.GetWidth()`) e per tutta la sua altezza (`Image.GetHeight()`).
`pChar` sarà un puntatore ti dipo unsigned char che conterrà l'indirizzo di ogni singolo pixel. Il valore del pixel puntato da `pChar` viene infine copiato all'interno della corrispondente posizione nella matrice.
Lo stesso procedimento si applica al passaggio inverso, dalla matrice alla struttura CImage, invertendo l'ultima istruzione.
Nel caso di immagini a 24 bit, al posto di una matrice di unsigned char se ne utilizza una di long e anche `pChar` è sostituito da `pLong`, un puntatore a long.
Se si necessita di scomporre l'immagine RGB nelle sue componenti si possono utilizzare le funzioni

```
pImageR[row][col]=GetRValue(pImage[row][col]);
pImageG[row][col]=GetGValue(pImage[row][col]);
pImageB[row][col]=GetBValue(pImage[row][col]);
```

in questo modo si inseriscono nelle tre matrici a 8 bit (`pImageR, pImageG, pImageB`) la rispettive componenti di colore prese dall'immagine a 24 bit.
Riunire le tre componenti nell'immagine originale è eseguito dall'istruzione

```
pImage[row][col]=RGB(pImageR[row][col],pImageG[row][col],pImageB[row][col]);
```

Una volta creata la (o le) matrice con i valori di colore dei pixel si opera su di essi come spiegato nel corso della tesi, applicando gli algoritmi descritti.

Manuale del Software

Questo manuale ad alto livello del software prodotto nel corso della tesi può risultare utile durante la consultazione del codice.
Abbiamo già introdotto che lo strumento utilizato per la realizzazione del software è stato Visual Studio 2005. Questo permette di creare delle finestre di lavoro pre-impostate su cui poter operare, grazie alla MFC (Microsoft Fundation Class).
Il programma si articola su dieci classi principali **CImageProcApp**, **CImageProcDoc**, **CImageProcView**, **CMainFrame**, **CAboutDlg**, **MemClass**, **CAnisotropicDiffusion**, **ImgProc, TesiTrieste**, **Prove**.
Le prime cinque sono generate in maniera automatica da Visual Studio e servono per la gestione della finestra di lavoro e degli eventi ad essa correlati.

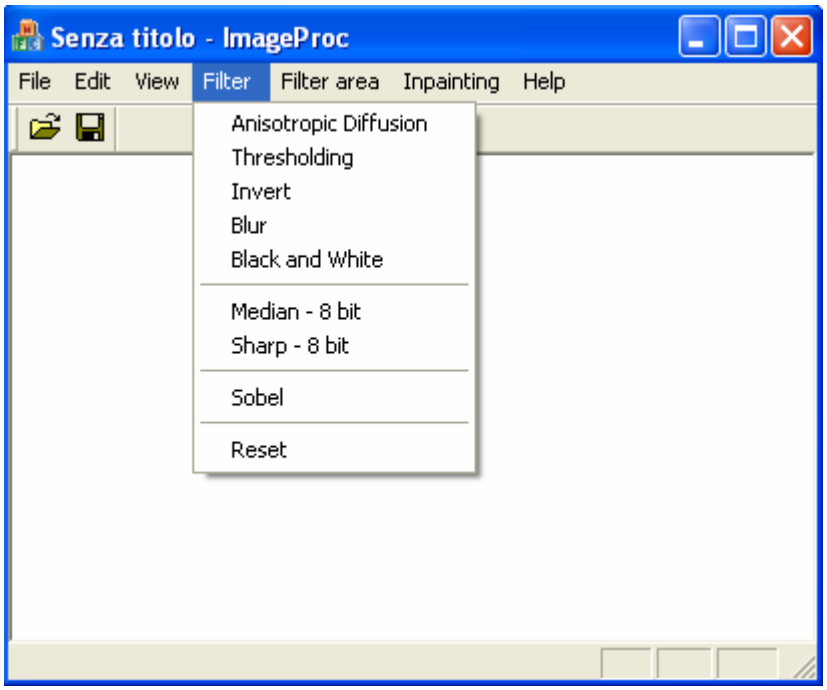


Fig M.1 - Come si presenta la finestra principale di lavoro

Nella figura M.1 si vede come appare la finestra principale del programma quando è attivato il menu “Filter”. Questo, come tutti gli altri menu, è gestito dalla classe **CImageProcView**, che lega la relativa voce del menu alla corrispondente gestore dell’evento. Il procedimento è eseguito tramite funzioni, per esempio, la voce “Blur” del menu “Filter” è legata all’evento **OnFilterBlur** che viene chiamata quando è selezionata la corrispondente voce del menu.
Lo stesso procedimento viene ripetuto per tutto il menu.
Schematizzando :

File → Open	OnFileApri
File → Salva	OnFileSalve
Filter → Invert	OnFilterInvert
Filter → Black and White	OnFilterBW
Inpainting → FDII	OnTriesteFDII
Inpainting → Texture	OnInpaintingTexture
.	.
.	.
.	.

e così via.Inoltre, sempre nella classe **CImageProcView**, sono contenute le funzioni **CImage08Bit2UnsignedChar**, **CImage24Bit2Long**, **UnsignedChar2CImage08Bit** e **Long2CImage24Bit**.
La prima trasforma l’immagine contenuta nella struttura CImage in una matrice di Unsigned Char, in cui ogni elemento corrisponde al valore del pixel alla posizione indicata dagli indici della matrice.
Stesso discorso per la seconda funzione, che invece di passare da CImage a Unsigned Char effettua la trasformazione CImage – Long.
La prima è usata per immagini a 8 bit mentre la seconda per spazi di colore a 24 bit (RGB).

Le funzioni **UnsignedChar2CImage08Bit** e **Long2CImage24Bit** computano l'operazione inversa, passando da una matrice di valori alla classe CImage. Questo passaggio è indispensabile per poter visualizzare a video l'immagine utilizzando la funzione **Draw** fornita proprio dalla CImage e eseguita all'interno dell'evento **OnDraw** della classe **CImageProcView** .
In tale funzione l'unico controllo necessario è sul numero di immagini disegnate nella finestra, come mostrato nella figura M.2.

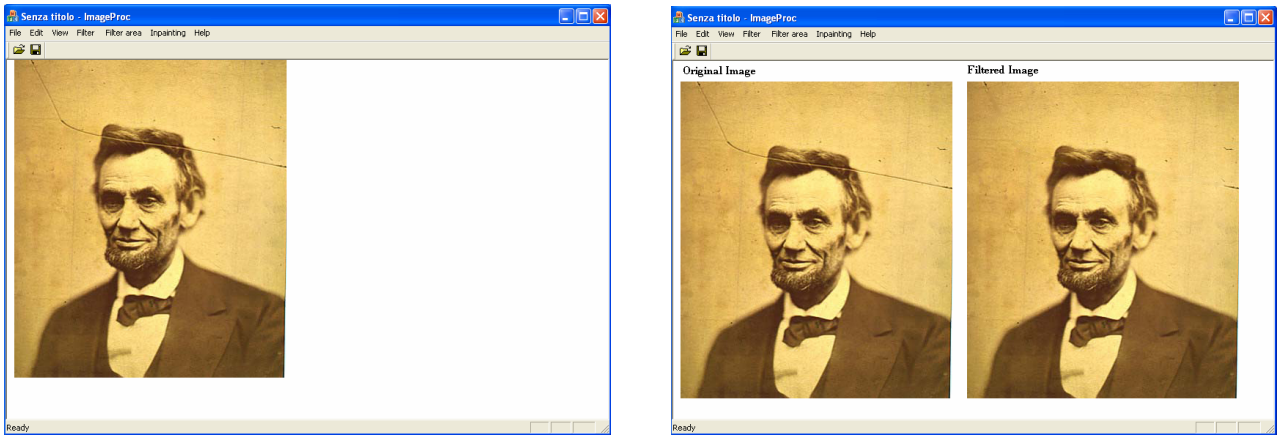


Fig. M.2 – A sinistra l'immagine originale da filtrate. A destra si hanno, nella stessa finestra, sia l"immagine originale che l'immagine filtrata

Possiamo avere una sola immagine, prima di effettuare una delle operazioni di filtraggio, oppure due immagini, l'originale e la filtrata.
Infine, sempre nella classe **CImageProcView**, vi sono i gestori di apertura e salvataggio dei file di immagine (BMP), **OnFileApri** e **OnFileSalva**.
Non si darà alcuna descrizione delle altre classi per la gestione delle finestra di lavoro (**CImageProcApp**, **CImageProcDoc**, **CMainFrame**, **CaboutDlg**), poiché non sono state minimamente toccate durante lo sviluppo del software.
La classe **MemClass** ha come utilizzo primario l'allocazione e la deallocazione dinamica di memoria, nel caso specifico permette di creare aree di memoria da adibire all'allocazione delle matrici contenenti i pixel dell'immagine, questo è

realizzato con la funzione **Data2dAllocation**, mentre con **Data2dDeallocation** si libera la memoria precedentemente allocata.

ImgProc è una classe di prova, contenente i primi passi basilari di semplice filtraggio di immagini, realizzata inizialmente in modo da prendere confidenza con l'elaborazione di immagini.

Per fare un esempio, le funzioni **Invert** e **Invert24** applicano l'inversione dei valori dei pixel di un immagine in ingresso (pixel_invertito = 255 – pixel), otenendone il negativo.

Oppure **Blur** e **Blur24**, che filtrano l'immagine calcolando nuovi valori per ogni pixel in base ai pixel che lo circondano, tramite una semplice media, ottenendo un effetto "sfocamento" sull'immagine in uscita.

Thresholding e **Thresholding24** mantengono solamente i valori di colori superiori o inferiori ad una soglia impostabile a piacere (ovviamente nell'intervallo 0-255), ponendo a "bianco" (o a qualsiasi altro colore) i restanti pixel. Risulta molto utile nella creazione delle maschere. Si pone la soglia in modo da "lasciar passare" al filtro solamente il colore nero, si disegna la maschera sull'immagine originale (anche con Paint di windows), utilizzando un pennello nero. A questo punto si può creare l'immagine maschera con la funzione Thresholding del menu Filter.

Ciò che si ottiene è proprio un'immagine con le caratteristiche desiderate, nera lungo il difetto e bianca nel resto dell'immagine.

Un tipo di filtro più avanzato è realizzato dalla classe **CAnisotropicDiffusion**.

La diffusione anisotropica effettua uno sfocamento adattativo dell'immagine, diffondendo il colore all'interno di aree uniformi e mantenendo nitidi i contorni, come spiegato nel capitolo 9 durante l'esposizione dell'algoritmo "Image Inpainting".

Le funzioni basilari sono **AD_24** e **AD**. La prima non fa altro che scomporre l'immagine a 24 bit nelle sue tre componenti di colore (chiamando la funzione **AD_LongRGB2UnsignedChar** sull'immagine) e richiamare la seconda funzione

su ognuna di esse. **AD** esegue l'algoritmo di diffusione anisotropica, computando le equazioni riportate a pagina 47. Lì possiamo notare come le due funzioni *g* dipendano solamente dai valori dei pixel dell'immagine, poiché *K* è una costante fissata dall'utente, ma i pixel assumono valori nell'intervallo 0-255, per cui è facilmente impostabile una tabella di look-up contenente tutti i possibili valori delle funzioni *g* in base al pixel in ingresso.
La tabella è creata dalla funzione **AD_LookUpTable**.
I valori della costante *K* possono essere cambiati accedendo al file AD.ini. Questo file, oltre a K, permette di impostare anche il numero di cicli di esecuzione dell'algoritmo (ovvero quante volte applicarlo ripetutamente all'immagine), il valore di λ (che ricordiamo essere vincolata: $0 \leq \lambda \leq 1/4$) e il tipo di funzione *g* desiderata (se gtype è impostato a 1 si esegue la funzione con l'esponziale, altrimenti l'altra).
Le tecniche di inpainting vere e proprie, sviluppate nel corso della tesi, sono contenute all'interno delle due restanti classi: **TesiTrieste** e **Prove**.
La prima delle due contiene l'algoritmo Fast Digital Image Inpainting [9] (cap. 9) e l'algoritmo di Marco Ortolan per il restauro di crepe [15] (cap. 11).
L' algoritmo FDII è gestito dalle due funzioni in cast **FDII**, una operante su immagini a 8 bit e l'altra su quelle a 24. Quest'ultima, diversamente da quanto visto prima per la diffusione anisotropa su immagini RGB, in cui si chiamava la funzione a 8 bit per ogni livello di colore, effettua un controllo interno.
L'idea di base consiste nel fatto che anche immagini in bianco e nero possono essere a 24 bit. In questo caso ognuna della componenti ha valore identico alle restanti. Diviene quindi inutile chiamare la funzione a 8 bit su tutte e tre i livelli di colore, è sufficiente lavorare su uno dei tre e copiarne succesivamente il risultato anche alle altre due componenti. Diversamente, con le immagini a colori, è necessario richiamare la funzione a 8 bit su ognuna delle componenti. Per farlo si passano all'altra funzione **FDII** (quella a 8 bit) , uno alla volta, tutti e tre i colori.

Questa funzione, come indicato dall'algoritmo, comincia con l'annullamento del colore all'interno del difetto, ovvero "sbiancando" l'area coperta dalla maschera. Dopodichè, per un numero di volte pari a **FDII_NUMBEROFCYCLES**, si esegue la convoluzione dei pixel presenti nell'area da restaurare, prima con un kernel e poi con l'altro. Ciò è portato a termine dalle due funzioni **FDIIFilterPixelKernelOne** e **FDIIFilterPixelKernelTwo**.

L'altro algoritmo contenuto nella classe **TesiTrieste**, proposto nella tesi di Marco Ortolan, non lavora con immagini a colori, ma solamente con immagini in scala di grigi, tuttavia la funzione principale, chiamata appunto **Ortolan**, riceve in ingresso una matrice di tipo long. Questo è dovuto al fatto che, arrivati a questo punto del lavoro di tesi, dedicato al restauro di immagini tratte da fotografie, ci si è accorti che nella stragrande maggioranza dei casai queste erano a 24 bit, anche se in bianco e nero. Si è tralasciato per semplicità di darne una versione operante a 8 bit. Quindi, la funzione **Ortolan** conta il numero dei pixel coperti dalla maschera e finchè tutti non sono stati visitati chiama la funzione **Ortolan_FCCN**. Questa esegue la Funzione di Cross Correlazione Normalizzata, come indicato nel capitolo relativo di questa tesi. La FCCN è eseguita sui pixel coperti dalla maschera ma non coperti dalle barriere e solo sui punti di bordo maschera. Poiché la matrice di ricerca della massima FCCN, ha forma dipendente dalla zona di posizionamento del pixel indagato relativamente alla maschera, è necessario distinguere se il pixel
appartiene al bordo superiore della maschera oppure a quello inferiore, piuttosto che a uno dei due laterali. La distinzione è applicata esaminando il pixel tramite le funzioni **IsUpMask**, **IsDownMask**, **IsLeftMask**, **IsRightMask**.

Al capitolo 11, leggendo la descrizione dell'algoritmo di Ortolan, si nota che questo opera solo sui pixel il cui valore dista dalla media dei vicini del pixel alla stessa posizione di quello in esame ma appartenente all'immagine filtrata con FDII, più di una certa quantitaà.

Questo controllo è eseguito dalla riga di codice:

```
if(Magnitude(pImage[row][col]-Ortolan_MeanFDII(pFDII,row,col))<FCCN_BETA)
```

Notiamo altre due funzioni, **Magnitude**, che calcola il modulo dell'argomento e **Ortolan_MeanFDII** che computa la media dei vicini, sull'immagine ottenuta da FDII a partire dall'originale.
Se il controllo risulta falso si calcola la FCCN. Durante questo calcolo si mantengono i valori del pixel risultato migliore fino a quel momento, ovvero quello con la più alta FCCN ricavata. Alla fine della ricerca si avranno quindi a disposizione gli indici corrispondenti alla posizione del pixel da sostituire a quello in esame. Prima di completare la sostituzione però bisogna controllare che il pixel ottenuto appartenga ad una corretta zona dell'immagine. Il controllo è contenuto nella funzione **Ortolan_ControlZone**.
Terminata l'esecuzione della FCCN per ogni pixel della maschera si ritorna alla funzione **Ortolan**.
Il passo successivo consiste nella chiamata alla procedura **Variance_Priority**.
Il suo compito risiede nella ricerca del pixel a più alta priorità tra quelli restanti, quelli che non hanno trovato un valido sostituto durante il computo della FCCN.
La priorità si ottiene tramite calcolo della varianza di una matrice di dimensioni 7x7 centrata nel pixel in esame, grazie alla funzione **Variance_Variance7x7**.
Il computo della varianza necessita la conoscenza della media, ottenuta dalla funzione **Variance_Mean7x7**.
Trovato il pixel a priorità massima, lo si sostituisce con la media dei pixel che lo circondano. Esistono tre diverse tipologie per il calcolo di questa media, in base al valore di varianza ottenuto per quel pixel, come indicato in [15].
La distinzione si ottiene tramite le tre funzioni **Variance_LowVariance**, **Variance_MediumVariance**, **Variance_HighVariance**.
Rimane da compiere l'ultimo passo, restaurare i pixel coperti dalla maschera delle barriere. Si computa un'altra volta la priorità tramite la solita **Variance_Priority**, senza però riaggiornare i valori di priorità ad ogni

sostituzione. E si opera come predentemente escplicato, tramite sostituzione con valor medio.

Queste ultime operazioni sono eseguite all'interno di **VarianceBarrier**.

L'altra classe contenente materiale relativo all'inpainting è **Prove**.

L'algoritmo di sintesi della tessitura a singola risoluzione, [10], spiegato nel capitolo 10, si basa sulle due funzioni di nome **Texture**.

In questo caso si è scelto di tornare alle "origini", realizzando una funzione per immagini a 8 bit, **Texture**, e una per funzione 24 bit, **TextureRGB**. Anche qui sulle immagini in bianco e nero a 24 bit si opera su una sola delle tre componenti.

Quindi, riassumendo, il gestore degli eventi contenuto nella classe **ImageProcView** (**OnInpaintingProvaTexture**), chiama una delle due funzioni **Texture** contenute nella classe **Prove**, in base al numero di bit utilizzati dall'immagine per ogni pixel. Se chiama quella che opera a 24 bit si necessita di un ulteriore controllo: se l'immagine è in bianco e nero viene passata una delle tre componenti all'altra procedura **Texture**, quella a 8 bit. Altrimenti si passano tutte e tre le componenti alla funzione **TextureRGB**.

Queste ultime due operano nello stesso modo, cercando, per ogni pixel sito lungo il bordo della maschera, un sostituto valido tramite il computo della SSD minima.

La differenza sostanziale sta proprio nel calcolo di questa differenza: nelle immagini in scala di grigi tiene conto di unico valore unsigned char per ogni pixel, contenuto nell'unica matrice pImage, mentre nelle immagini RGB è necessario calcolare la differenza nel seguente modo:

$$D(i,j) = \left[I_R(i,j) - I_R^{'}(i,j) \right]^2 + \left[I_G(i,j) - I_G^{'}(i,j) \right]^2 + \left[I_B(i,j) - I_B^{'}(i,j) \right]^2$$

generando così due funzioni distinte, **SSD** e **SSD_RGB**.

Stesso identico discorso può essere fatto per le funzioni **TexturePezza** e **TexturePezzaRGB**. Queste due applicano lo stesso algoritmo di **Texture** e **TextureRGB**, utilizzando ancora **SSD** e **SSD_RGB** nella ricerca di un sostituto valido da inserire al posto del difettoso. L'unica differenza sta proprio nella sostituzione. Anziché copiare il singolo pixel si effettua una copia di un'intera "pezza", consistente in una matrice di dimensioni variabile (a scelta dall'utente). Questa tecnica, congiuntamente alla priorità, è introdotta da Criminisi et al. nell'algoritmo [12].
Il funzionamento della priorità è mostrato tramite le funzioni **TexturePriorita**, implementata più che altro con il fine di mostrare in quali punti dell'immagine va ad applicarsi questa priorità.
La funzione **Texture** lavora sull'argotmo di Wei e Levoy a singola risoluzione.
La multirisoluzione è introdotta con le funzioni **TexturePiramide** (8 e 24 bit) e **TexturePiramideRGB**. Ancora una volta si fa la distinzione tra immagine a 8 e 24 bit in bianco e nero e 24 bit a colore.
TexturePiramide inizia con la creazione del primo livello della piramide Gaussiana, tramite media pesata di quattro pixel vicini, poi crea il primo livello della piramide Laplaciana per sottrazione del filtraggio gaussiano dall'immagine originale.
Dal primo livello si calcola il secondo livello, mediando su sedici pixel adiacenti e ancora si calcola per sottrazione il secondo livello Laplaciano.
A questo punto viene applicato l'algoritmo di sintesi di tessitura a singola risoluzione sull'ultimo livello creato, con **TexturePirAlgPrimo**.
Successivamente si ricrea il primo livello, con i nuovi dati ottenuti dal restauro al livello inferiore, e si chiama **TexturePirAlg**, che, come spiegato nel capitolo 10, differisce da **TexturePirAlgPrimo** poiché sfrutta i risultati ottenuti ai livelli inferiori. Dall'output di **TexturePirAlgPrimo** si crea nuovamente l'immagine originale e le si applica per l'ultima volta **TexturePirAlg**.

Nel capitolo 12 si parla di scomposizione dell'immagine in modo da utilizzare combinatamente sintesi della tessitura e diffusione, in modo particolare si tratta l'algoritmo [17]. Questo è implementato con le funzioni **Yamauchi**.
Come già introdotto, per via della sua estrema lentezza, dovuta al computo della DCT, ne è stata implementata una versione solamente per immagini in bianco e nero, a 8 e 24 bit.
A grandi linee: si applica la DCT sull'immagine, proprio con la funzione **DCT**, questa ritorna però solo la componente dell'immagine alle basse frequenze.
Per ottenere ciò pone a zero tutti i coefficienti delle bande alle alte frequenze all'interno della matrice contenente i valori della DCT.
Tramite sottrazione dell'immagine a bassa frequenza dall'immagine originale si ottiene la parte con le alte frequenze.
A questo punto si applica l'agoritmo di FDII sulla prima e l'algoritmo di texture sulla seconda, chiamando le apposite funzioni e infine si ricrea l'immagine originale sommando i risultati ottenuti.

Manuale di utilizzo del software

Come prima cosa ricordiamo che tutti gli algoritmi implementati sono applicabili solamente a immagini di tipo bitmap (.bmp).

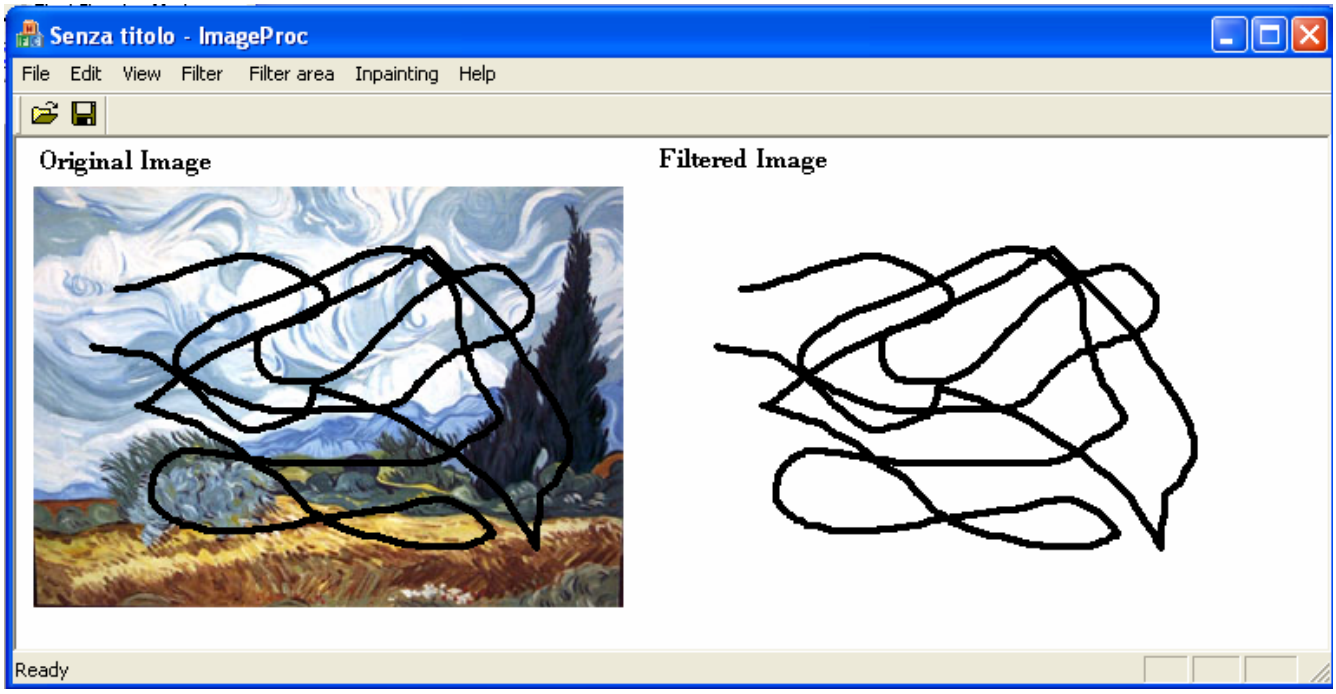
La cartella contenente il file eseguibile (ImageProc.exe) deve contenere anche la cartella **IMAGES**, quest'ultima avrà al suo interno altre tre directory: **Image**, **Mask** e **Barrier**, nella prima collocheremo le immagine difettose da ripare, nella seconda le maschere identificanti il difetto e nella terza le barriere, se necessario.

Inoltre sarà necessario mantenere un'altra cartella allo stesso livello dell'eseguibile, denominata **INI**, all'interno della quale verrà posizionato il file **AD.ini**, indispensabile per l'utilizzo della diffusione anisotropica.

Le maschere e le barriere sono create nello stesso identico modo: si apre l'immagine difettosa con un comunissimo programma di disegno (per esempio Paint di windows) e con il pennello nero si colora l'area da restaurare (o le zone su cui applicare il meccanismo delle barriere) e si salva l'immagine con un nome diverso. Se l'immagine si chiamava, per esempio, img.bmp ed era contenuta in IMAGES/Image, possiamo chiamare la maschera img_mask.bmp e salvarla in IMAGES/Mask, oppure se si tratta di una immagine di barriere la si può chiamare img_bar.bmp e salvarla in IMAGES/Barrier.

Questa operazione non è indispensabile, ma facilita l'utilizzo dell'applicazione, poiché permette di aprire in modo automatico le cartelle relative a immagini, maschere e barriere quando richiesto, anziché andarle a cercare manualmente per tutta la memoria.

A questo punto è necessario modificare le maschere in modo da ottenerne una parte nera (che identifica il difetto o le barriere) e tutto il resto bianca. L'operazione è semplicemente attuabile proprio con il software realizzato. Una volta eseguito ImageProc.exe si apre la maschera selezionata (es. img_mask.bmp) e si seleziona la voce **Thresholding** dal menu **Filter**, il risultato sarà proprio ciò che ci si aspetta, come mostrato in figura



Ora è necessario risalvare l'immagine con lo stesso nome, scegliendo **Salva** dal menu **File**. Rimane un'ultima cosa da eseguire: l'immagine così ottenuta è a 24 bit, ma poiché comprende solo i colori bianco e nero sarebbe un'inutile spreco mantenerla tale, per cui occorrerà riaprire l'immagine con Paint e salvarla a 8 bit (**File** -> **Salva con nome** impostando il tipo di file come "Bitmap a 256 colori").

Vediamo ora le operazioni fondamentali per filtare le immagini.

Per prima cosa apriamo una delle immagini difettose (**File** -> **Apri**). Ora abbiamo a disposizione gli algoritmi di inpainting dell'omonima voce del menu. Tutti quanti ci chiederanno di aprire una maschera, fondamentale all'algoritmo per poter operare.

Gli algoritmo FDII e Ortolan necessitano ulteriormente delle barriere.

Quest'ultimo algoritmo utilizza l'output di Fast Digital Image Inpainting come riferimento, per cui, nel caso in cui io voglia filtrare l'immagine con Ortolan devo:

- aprire l'immagine
- selezionare **FDII** dal menu **Inpainting**

- scegliere una maschera e un'immagine con le barriere
- selezionare **Ortolan** dal menu **Inpainting**
- scegliere una maschera e una immagine con le barriere.

Ogni volta che vorrò annullare tutte le operazioni eseguite fino a quel momento su un'immagine mi basterà selezionare la voce **Reset** dal menu **Inpainting**.
Per quanto riguarda le semplici operazioni di filtraggio del menu **Filter**, nessuna di esse richiede maschere o input aggiuntivi oltre all'immagine iniziale, quindi basterà aprire un'immagine e scegliere una delle voci del menu.
Il menu **Filter Area** esegue tre operazioni di filtraggio (**Invert**, **Blur** e **Anisotropic Diffusion**) su un'area selezionata tramite mouse dell'immagine. Quindi: apro l'immagine, seleziono una delle tre voci di questo menu, clicco sull'immagine e, tenendo premuto il tasto, mi sposto lungo la sua superficie.
Anche le operazione eseguite con **Filter** e **Filter Area** sono removibili, selezionando **Filter** -> **Reset**.

Bibliografia

[1]
03_StancoRamponi_Towards the Automated Restoration of Old Photographic Prints A Survey

[2]
“Principi dell’IFLA per la cura ed il trattamento dei materiali di biblioteca.”
a cura di Edward P. Adcock con la collaborazione di Marie-Thérèse Varlamoff e Virginie Kremp traduzione di Luciano Carcereri e Rosa Martucci.
International Federation Of Library Associations And Instruction
Core Programme On Preservation And Conservation
Council On Library And Information Resources

[3]
”Digital Image Processing Techniques for the Detection and Removal of Cracks in Digitized Paintings”
Ioannis Giakoumis
IEEE Transactions On Image Processing, VOL. 15, NO. 1, January 2006

[4]
“Iterative image transforms for an automatic screening of cervical smears,”
F. Meyer
J. Histoch. Cytochem., vol. 27, pp. 128–135, 1979.

[5]
“Median radial basis function neural network,”
A. G. Bors and I. Pitas
IEEE Trans. Neural Netw., vol. 7, no. 6, pp. 1351–1364, Nov. 1996.

[6]
”Digital Image Processing Techniques for the Detection and Removal of Cracks in Digitized Paintings”
Ioannis Giakoumis
IEEE Transactions On Image Processing, VOL. 15, NO. 1, January 2006

[7]
“Image Inpainting”
M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester,
Computer Graphics (SIGGRAPH 2000),
pp. 417–424, July 2000.

[8]
“Scale-Space and Edge Detection Using Anisotropic Diffusione”
Pietro Perona, Jitendra Malik
IEEE Transactions On Pattern Analysis and Machine Intelligence, VOL.12, NO.7, July 1990

[9]
“Fast Digital Image Inpainting”
Manuel M. Oliveira, Brian Bowen, Richard McKenna, Yu-Sung Chang
Proceedings of the International Conference on Visualization, Imaging and Image Processing
September 2001

[10]
"Fast Texture Synthesis using Tree-Structured Vector Quantization"
Li-Yi Wei, Marc Levoy
Proceedings of SIGGRAPH 2000

[11]
"Pyramid Methods in Image Processing"
E.H.Adelson, C.H.Anderson, J.R.Bergen, P.J.Burt, J.M.Ogden
RCA Engineer, December 1984

[12]
"Object Removal by Exemplar-Based Inpainting"
Antonio Criminisi, Patrick Perez, Kentaro Toyama
WI Proc. IEEE Computer Vision and Pattern Recognition, June 2003

[13]
"Digital Inpainting – Survey and Multilayer Image Inpainting Algorithms"
Timothy K. Shih, Rong-Chi Chang
Proceedings of the Third International Conference on Information
Technology and Applications (ICITA '05) 2005 IEEE

[14]
"Probability and Statistical Inference"
Robert Bartoszynski, Magdalena Newiadomska-Bugaj
A Wiley-Interscience Publication, 1996

[15]
"Tecniche Digitali per il Restauro Virtuale di Stampe Fotografiche Affette da Danni Meccanici"
Marco Ortolan
Università degli Studi di Trieste, Facoltà di Ingegneria
A.A. 2004-2005

[16]
http://it.wikipedia.org/wiki/Fotografia

[17]
"Image Restoration using Multiresolution Texture Synthesis and Image Inpainting"
Hitoshi Yamauchi, Jorg Haber, Hans-Peter Seidel
Proc. Computer Graphics International (CGI) July 2003

[18]
"Fundamentals of Electronic Image Processing"
Arthur R. Weeks, Jr.
SPIE/IEEE series on imaging science & engineering, 1996

[19]
"Digital Image Processing"
R.C. Gonzalez, R.E. Woods
Second Edition, Prentice Hall, 2002

[20]
"Synthesizing Natural Textures"
Michael Ashikhmin
The Proceedings of 2001 ASM Symposium on Interactive 3D Graphics

[22]
"Simultaneous Structure and Texture Image Inpainting"
M.Bertalmio, L.Vese, G.Sapiro, S.Osher
Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)

[23]
"Teoria e Tecnica del Trattamento delle Immagini"
Andrea Pizzirani
Università degli Studi di Ferrara

[24]
"Algorithms Regarding Automatic Retouching of User Selected Regions in Digital Images"
Samuel Adolfsson
Department of Numerical Analysis and Computer Science
Royal Institute of Technology, SE-100 44 Stockholm, Sweden

Ringraziamenti

A conclusione di questa tesi di laurea desidero ringraziare il Professor Alberto Borghese per avermi dato la possibilità di svolgere questa tesi. Inoltre desidero ringraziarlo per avermi assistito e indirizzato al meglio per poter giungere a conclusione di questo lavoro e dal punto di vista umano, per aver compreso alcuni dei miei problemi sorti nel periodo di tesi.
Vorre ringraziare anche il Dottor Iuri Frosio che nei momenti di crisi "tecnica" ha sempre saputo darmi il giusto consiglio e la giusta imboccato per venirne fuori al meglio.
Infine vorrei ringraziare l'Ingegnere Marco Ortolan, che con disponibilità mi ha aiutato oltre i proprio doveri in un passo importante del mio ragionamento.