

POLITECNICO DI TORINO

Corso di Laurea
in Matematica per l'Ingegneria

Tesi di Laurea

Due metodi numerici per il filtraggio di immagini digitali



Relatori

prof. Adriano Festa
firma del relatore

.....

Candidato

Raffaello Ippolito
firma del candidato

.....

Anno Accademico 2021-2022

All'autrice di "Passi" ♡

Sommario

Le immagini sono da sempre state fondamentali per l'uomo, che sin dall'antichità le ha utilizzate per ricordare, illustrare, comunicare, etc. In tempi più recenti, questo fenomeno ha acquistato progressivamente centralità nella società contemporanea, spesso definita 'società dell'immagine'. La manipolazione, l'archiviazione e il trasferimento di contenuto attraverso le immagini, è una attività costantemente praticata e, per questo motivo, lo sviluppo di tecniche sofisticate per tali scopi è un problema di grande interesse in vari campi della scienza e della tecnica. In questo lavoro ci occuperemo principalmente di tecniche digitali di filtraggio. Lo scopo di tali tecniche è modificare l'immagine, estraendone elementi, nascondendoli o facendoli risaltare, così da migliorarla per renderla quanto più utile possibile agli scopi richiesti.

Durante questa trattazione, dopo una breve introduzione volta a fornire una visione d'insieme del campo in esame, verrà illustrato il concetto di immagine digitale, di filtro e di equazioni alle derivate parziali, con particolare riguardo al ruolo fondamentale che queste ultime giocano nella scrittura dei filtri e la relativa implementazione all'interno del calcolatore.

Nello specifico, nella parte introduttiva si parlerà della storia delle immagini digitali e del filtraggio analogico e digitale, dei principali problemi consequenziali e di alcune delle più semplici elaborazioni digitali. Verranno poi fornite alcune nozioni preliminari tra cui: il concetto di immagine come funzione matematica e quindi di immagine digitale, della relativa codifica, di pixel, del concetto di risoluzione ed altresì di quanto essa influisca sullo spazio di archiviazione. Successivamente saranno analizzati i filtri intesi come convoluzioni tra la funzione immagine e la funzione filtro.

Infine la trattazione diventerà più specifica, presentando il fulcro di questo studio, ossia la presentazione di un metodo di filtraggio anisotropico basato sulle equazioni alle derivate parziali comunemente chiamato metodo Perona Malik. A sostegno di ciò verrà illustrato il metodo delle differenze finite per l'approssimazione delle derivate parziali con relativa analisi dell'errore di troncamento. Questo risultato sarà dunque sfruttato per la realizzazione di uno script MATLAB che operi l'equazione del calore e che verrà poi opportunamente modificato per implementare il metodo Perona-Malik.

Ringraziamenti

Ringrazio vivamente il Politecnico di Torino con i suoi docenti e lavoratori per i servizi offerti nel corso di questi anni, ma soprattutto ringrazio la mia famiglia che mi ha sostenuto, permettendomi di vivere qui a Torino e di frequentare il Politecnico affinché io potessi vivere la miglior esperienza universitaria possibile.

Ringrazio inoltre i miei coinquilini: Pierclaudio, con cui condivido il tetto fin da quando sono arrivato qui a Torino, prima in collegio, poi in Via san Paolo e tutt'ora in Corso Ferrucci, dove ho trovato Francesco e Mattia che nonostante lo scarso tempo da me trascorso in casa sono stati testimoni dei miei scleri riguardo la stesura delle pagine di questo elaborato e non solo. Ringrazio le persone che mi sono state vicine, in particolare Amira che mi ha sostenuto più di chiunque altro in questo ultimo anno, subendo le mie lamentele riguardo agli esami, incoraggiandomi e confortandomi quando le cose non sono andate come sperato. Ringrazio le persone che mi sono state vicine fin dal mio arrivo a Torino: Francesco, Marco, Ginevra e poi Gigi, Michele, Nara, Enrica e Serena con cui ho condiviso tantissimo tempo, tanto in sala studio quanto al di fuori. Ringrazio chi c'è sempre stato, gli amici che ho lasciato al sud e che nonostante il tempo mi hanno sempre aspettato e riaccolto ad ogni mio ritorno, in particolare Luigi e l'amicizia che ci lega ormai da più di 10 anni, Giovanna con cui ho vissuto esperienze di ogni tipo e Silvana che fin da quando ci siamo conosciuti non mi ha abbandonato per un solo giorno e con cui ho il piacere di condividere il giorno della laurea, sebbene questo ci impedisca di partecipare l'uno alla proclamazione dell'altro. Ringrazio infine lo scoutismo ed il Torino 110 che mi ha accolto in questa città nuova per me e mi ha donato molti dei miei amici più cari e sopra citati. A tutti voi, grazie di cuore.

Indice

Elenco delle figure	8
1 Introduzione	11
1.1 Storia del filtraggio e principali problemi affrontati	11
1.1.1 Filtraggio analogico	11
1.1.2 Immagini digitali	12
1.1.3 Filtraggio digitale	13
1.2 Nozioni introduttive	14
1.2.1 Immagini digitali	14
1.2.2 Cosa sono i filtri	16
1.3 Principali problemi di elaborazione digitale	17
1.3.1 Rotazioni, riflessioni,etc	17
1.3.2 Cambio prospettiva	18
1.3.3 Morphing	19
2 Filtraggio digitale via Equazioni alle Derivate Parziali	21
2.1 Equazione del calore e la sua approssimazione numerica	21
2.1.1 Metodo delle differenze finite	21
2.1.2 L'equazione del calore	26
2.1.3 Studio della stabilità del metodo	27
2.2 Metodo Perona-Malik	30
2.2.1 Soluzione discreta della PDE	31
2.2.2 Maschere di convoluzione	32
2.2.3 Rilevamento dei bordi	35
2.2.4 Problema dei " <i>bordi</i> " d'interferenza	41
2.2.5 Implementazione	42
3 Esempi di utilizzo	45
3.1 Esempi di utilizzo	45
3.1.1 Esempio 1 - Risultati ottenuti con i parametri standard	46
3.1.2 Esempio 2 - Variazione del numero di iterazioni	47
3.1.3 Esempio 3 - Variazione della costante d'integrazione	49
3.1.4 Esempio 4 - Variazione della costante di controllo	51
3.1.5 Esempio 5 - Variazione della costante di diffusione	54

3.2 Conclusioni	57
---------------------------	----

Elenco delle figure

1.1	Mascheratura per la correzione del contrasto. Fodde [2000]	12
1.2	Nastro dati codificante un’immagine.	12
1.3	La prima immagine risale al 1921 ed è composta da 5 gradazioni di grigio. La seconda è del 1929 ed è composta da 15 gradazioni di grigio. Gonzalez and Wood [1992]	13
1.4	Confronto tra immagine originale e immagine codificata utilizzando una griglia 4x4.	15
1.5	Confronto tra immagine originale e immagine codificata utilizzando una griglia 80x80.	15
1.6	Definizione dei punti e delle linee sull’immagine di partenza e sull’immagine d’arrivo Anton H. [2010] e uno schema che mostra la trasformazione tra lo stato iniziale a quello finale di un singolo triangolo con un passaggio intermedio	20
1.7	Processo di morphing con alcuni risultati intermedi.	20
2.1	Effetti dell’equazione del calore nel tempo.	27
2.2	Esempio di maschera di convoluzione 3x3	32
2.3	Griglia dell’immagine in nero, maschera in verde, pixel in esame in blu	33
2.4	Intorno di un punto, maschera di convoluzione e soluzione analitica	33
2.5	Derivate parziali di una tinta unita.	37
2.6	Derivate parziali, gradiente e laplaciano di una sfumatura orizzontale.	37
2.7	Derivate parziali, gradiente e laplaciano di una parziale sfumatura diagonale. .	38
2.8	Derivate parziali, gradiente e laplaciano di una figura semplice.	39
2.9	Derivate parziali, gradiente e laplaciano di una figura semplice con rumore. .	40
3.1	Da sinistra a destra: immagine con rumore, immagine filtrata con equazione del calore e immagine filtrata con metodo Perona-Malik.	46
3.2	In ordine: immagine con rumore e immagine filtrata dopo 10 iterazioni, dopo 20 iterazioni e dopo 100 iterazioni.	47
3.3	In ordine: immagine con rumore e immagine filtrata dopo 10 iterazioni, dopo 20 iterazioni e dopo 100 iterazioni.	48
3.4	In ordine: immagine con rumore, immagine filtrata con $\delta_t=0.01$, con $\delta_t=0.1$ e con $\delta_t=0.25$	49
3.5	In ordine: immagine filtrata con $\delta_t=0.1$, e con $\delta_t=0.25$	50
3.6	Immagine filtrata con $\delta_t=0.3$	50
3.7	In ordine: immagine con rumore, immagine filtrata con $c=6$, con $c=20$ e con $c=60$	51

3.8 In ordine: immagine filtrata con $c=60000$ e immagine filtrata con equazione del calore.	52
3.9 In ordine: immagine con rumore e immagine filtrata con $c=0.001$	53
3.10 In ordine: immagine con rumore, immagine filtrata con $\sigma=0.5$, con $\sigma=1$ e con $\sigma=2$	54
3.11 In ordine: immagine con rumore, immagine filtrata con $\sigma=0.5$, con $\sigma=1$ e con $\sigma=2$	55
3.12 In ordine: immagine originale, immagine filtrata con equazione del calore e immagine filtrata con $\sigma=2$	56

Capitolo 1

Introduzione

1.1 Storia del filtraggio e principali problemi affrontati

L'essere umano, da sempre, sfrutta i propri sensi per relazionarsi con il mondo. Il suo primo riferimento in particolare è la vista, senza la quale egli si sente perso, smarrito; per questo è fondamentale sviluppare un linguaggio visivo che sia efficace. La società contemporanea sfrutta immagini di vario genere (fotografie, radiografie, ecografie, poster pubblicitari, progetti, etc.) per comunicare messaggi e/o per rendere chiari progetti, idee, situazioni. È quindi un problema sempre di grande interesse cercare di sfruttarle al meglio. A tal proposito esistono alcuni metodi di **filtraggio**, la cui finalità è quella di migliorare la qualità delle immagini, metterne in risalto determinate caratteristiche o nasconderne delle altre.

1.1.1 Filtraggio analogico

Anche prima dell'avvento dei computer esisteva l'elaborazione delle immagini. Esistevano infatti dei procedimenti, detti "*mascherature*", che servivano a ridurre o esaltare le differenze di luce nella foto.

La mascheratura avveniva durante la fase di stampa su carta, ossia quando il negativo veniva proiettato sulla carta fotografica mediante l'ingranditore. Qui, l'operatore utilizzava una serie di metodi per far sì che su certe zone della carta andasse più o meno luce di quella che sarebbe arrivata passando attraverso il negativo. Ad esempio la tecnica di mascheratura per la correzione del contrasto prevedeva la sovrapposizione a registro della stessa immagine sia in positivo che in negativo, con lo scopo di ridurre o accettuare il contrasto.



Figura 1.1. Mascheratura per la correzione del contrasto. [Fodde \[2000\]](#)

Partendo da questa si sono sviluppate tecniche specifiche come la mascheratura detta ad "*altissimo contrasto*" permetteva di avere foto con soli bianchi e soli neri, molto simile a quello che oggi chiamiamo "*posterizzazione*"

Il viraggio invece serviva a dare un tono di colore alla foto, che restava comunque un bianco e nero: famoso il viraggio tono seppia.

1.1.2 Immagini digitali

Le immagini digitali trovarono le prime applicazioni sui giornali negli anni 20¹. Non esistevano veri e propri computer, il segnale era trasmesso tramite telegrafo simulando dei mezzitoni. In particolare, il sistema di trasmissione di immagini via cavo Bartlane era una tecnica inventata nel 1920 per trasmettere immagini di giornali digitalizzate su linee sottomarine tra Londra e New York.

Il sistema di trasmissione di immagini via cavo Bartlane generava sia sul trasmettitore che sul ricevitore una scheda dati perforata o un nastro che veniva ricreato come immagine.

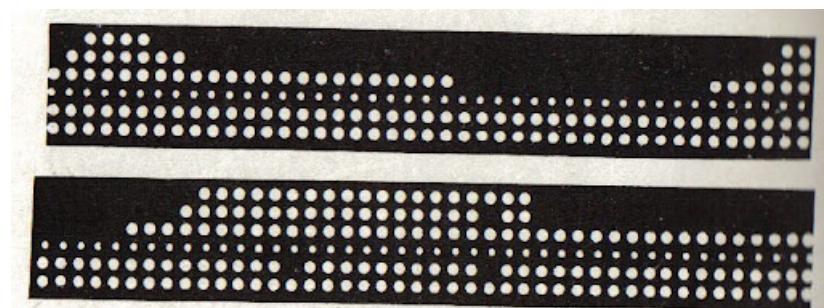


Figura 1.2. Nastro dati codificante un'immagine.

¹[Gonzalez and Wood \[1992\]](#)

In questo modo riusciva a codificare immagini in bianco e nero in cinque diverse gradazioni di grigio, capacità poi ampliata a 15 gradazioni nel 1929.



Figura 1.3. La prima immagine risale al 1921 ed è composta da 5 gradazioni di grigio. La seconda è del 1929 ed è composta da 15 gradazioni di grigio. [Gonzalez and Wood \[1992\]](#)

Questa, in qualche modo, è la nascita delle immagini digitali, anche se non sono trattate da computer e quindi codificate in maniera differente. Inoltre in questa fase non abbiamo una vera e propria elaborazione delle immagini, ma solo una trasmissione e la successiva stampa.

Nel 1957, Russell A. Kirsch produsse un dispositivo che generava dati digitali che potevano essere archiviati in un computer; questo utilizzava uno scanner a tamburo e un tubo fotomoltiplicatore. Negli anni immediatamente successivi, tale invenzione portò a notevoli sviluppi.

1.1.3 Filtraggio digitale

Negli anni 60 del XX secolo vari laboratori come il Jet Propulsion Laboratory, il Massachusetts Institute of Technology, i Bell Laboratories e l'Università del Maryland svilupparono molte tecniche di filtraggio digitale di immagini (o trasformazione di immagini digitali come spesso veniva chiamata) al fine di evitare le debolezze operative delle fotocamere a pellicola per missioni scientifiche e militari². Queste trovarono poi applicazione in immagini satellitari, immagini medici, videocitofono, riconoscimento ottico dei caratteri, e miglioramenti fotografici.

Il costo dell'elaborazione in quel periodo era piuttosto alto per via del prezzo elevato delle apparecchiature utilizzate. Le cose cambiarono negli anni settanta, quando sul mercato furono resi disponibili computer più economici e hardware dedicato.

Le immagini allora potevano essere elaborate in tempo reale, l'elaborazione digitale sostituì così i vecchi metodi a pellicola per molti scopi.

Negli anni 2000, grazie all'avvento di computer più veloci, il filtraggio digitale divenne la forma più comune di elaborazione delle immagini e, in generale, divenne il metodo più utilizzato data la sua versatilità e il basso costo. Le tecnologie utilizzate per il filtraggio delle immagini digitali per applicazioni mediche è stata inserita nel 1994 nella Hall of Fame della Space Foundation, esse infatti permisero di sviluppare la tecnica della tomografia computerizzata assiale, meglio nota come TAC molto utilizzata anche ai giorni nostri in radiologia.

²[Gonzalez and Wood \[1992\]](#)

1.2 Nozioni introduttive

La società moderna è una società digitale. Le immagini passano generalmente per un calcolatore prima di essere stampate, o in ogni caso possono essere sempre scannerizzate (con strumenti più o meno precisi) così da averne una copia digitale. È in questa fase che l'immagine subisce il processo di filtraggio, nel calcolatore, quando è in formato digitale. Per capire cos'è un filtro occorre dunque chiedersi cosa sia un'immagine digitale.

1.2.1 Immagini digitali

È necessario comprendere a fondo che cosa sia un oggetto per poterlo successivamente implementare in un calcolatore. A tal scopo è utile quindi capire esattamente che cos'è un'immagine

Forma esteriore degli oggetti corporei, in quanto viene percepita attraverso il senso della vista, o si riflette – come realmente è, o variamente alterata – in uno specchio, nell'acqua e sim., o rimane impressa in una lastra o pellicola o carta fotografica.

Vocabolario Treccani

Un'immagine viene rappresentata impressa su superfici, cioè oggetti bidimensionali, di dimensioni finite la cui visione è resa possibile attraverso i nostri occhi che percepiscono il susseguirsi di diversi colori. Come codificare tali entità? Come per tutti gli oggetti reali, sebbene questi ultimi abbiano dimensioni finite, le immagini sono distribuzioni continue di colore, o per meglio dire, il susseguirsi delle gradazioni di colore avviene in una maniera che possiamo considerare come continua. La soluzione più largamente utilizzata è anche quella più semplice ed intuitiva, ossia di discretizzare tale distribuzione di colori. Si divide l'immagine con una griglia e ad ogni casella, che d'ora in poi chiameremo **pixel**, assegnamo un colore. È ovvio che così facendo si perdono dei dettagli, la quantità di dettagli che riusciamo a conservare può variare enormemente, una minima quantità di dettagli si perde sempre ma è un prezzo che vale la pena pagare.

Facciamo un esempio:



Figura 1.4. Confronto tra immagine originale e immagine codificata utilizzando una griglia 4x4.

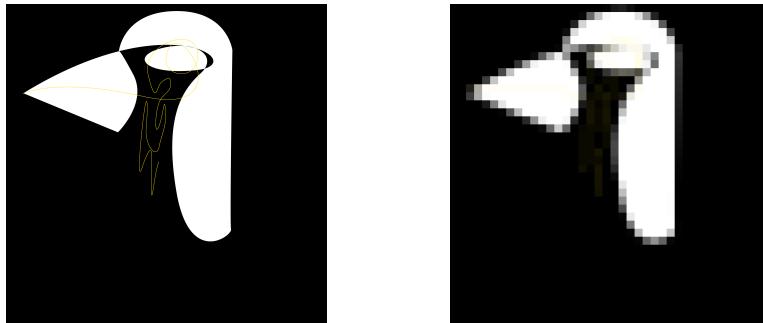


Figura 1.5. Confronto tra immagine originale e immagine codificata utilizzando una griglia 80x80.

È semplice vedere come ad una griglia più fitta corrisponda una miglior qualità dell'immagine, questo è il concetto di **risoluzione di un'immagine**. Una miglior risoluzione però costa in termini di memoria. Una griglia 4x4 corrisponde a 16 pixel, ad una griglia 80x80 corrispondono invece 6400 pixel! Questo vuol dire che la seconda immagine pesa 400 volte di più della prima. In particolare se l'immagine è in bianco e nero, per ogni pixel viene occupato 1 bit di memoria, bastano infatti 2 valori: 0 è nero, 1 è bianco. Per memorizzare un'immagine in scala di grigi invece occorrono 8 bit (ossia un byte) per pixel, abbiamo quindi a disposizione 256 valori 0 è nero, 255 è bianco e tutti i numeri naturali tra 1 e 254 rappresentano sfumature di grigio con 1 che rappresenta la gradazione di grigio più scura e 254 la più chiara. Tornando all'esempio capiamo che formattandola in bianco e nero la prima peserà 2 byte e la seconda 800 byte, formattata in scala di grigi la prima peserà 16 byte mentre la seconda 6.4 chilobyte. Da questo semplice calcolo capiamo quindi che cambiando tipo di formattazione, per conservare più dettagli, l'occupazione di memoria aumenta parecchio, soprattutto con immagini a grande risoluzione. A riprova di ciò osserviamo che mentre nel passaggio da una risoluzione di 4x4, ad una 80x80 mantenendo la stessa formattazione la seconda è 400 volte più pesante, nel passaggio invece da una risoluzione di 4x4 in bianco e nero ad una di 80x80 in scala di grigi, siccome per ogni pixel serve 1 byte

(cioè 8 volte di più di quanto occorre in bianco e nero), l'aumento sarà quindi $400 \cdot 8 = 3200$, cioè la seconda immagine peserà 3200 volte di più della prima.

Volendo definire in maniera più precisa che cosa è un'immagine digitale, diremmo che quest'ultima è una funzione $u : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ cioè, date in input due coordinate, essa restituisce un colore (che è formato da 3 canali RGB). Se però l'immagine è in bianco e nero la questione si semplifica: l'immagine diventa una funzione $u : \mathbb{R}^2 \rightarrow \mathbb{R}$, dal momento che per codificare un colore appartenente alla scala di grigio basta un solo canale: il livello di luminosità. Difatti, la riduzione da tre ad un solo canale rappresenta un grosso vantaggio, permettendo di diminuire di due terzi lo spazio di memoria occupato. In un'immagine a colori infatti ognuno dei 3 canali è codificato come per la scala di grigi ma sono invece "scale di rosso, blu e verde" questo vuol dire che per ogni pixel necessitiamo di 3 byte. Riprendendo ancora una volta l'esempio delle due immagini a bassa ed alta risoluzione, formattate come immagine a colori la prima peserà 48 byte mentre la seconda 19.2 chilobyte. Nel passaggio da una risoluzione di 4x4 in bianco e nero ad una di 80x80 a colori, siccome per ogni pixel servono 3 byte di memoria, l'aumento sarà $400 \cdot 8 \cdot 3 = 9600$, cioè la seconda immagine peserà 9600 volte di più della prima.

1.2.2 Cosa sono i filtri

Una volta capito che un'immagine è una funzione possiamo definire un filtro come una seconda funzione $u : \mathbb{R}^2 \rightarrow \mathbb{R}$ che convoluta alla prima da il risultato richiesto.

In matematica, la **convoluzione** è un'operazione tra due funzioni $f, g \in L^1(X)$ a valori in \mathbb{R} che consiste nell'integrare il prodotto tra la prima e la seconda traslata di un certo valore, in formule:

$$h = (f * g)(x) = \int_X f(y)g(x - y) dy = \int_X f(x - y)g(y) dy$$

Si dimostra che h è ancora una funzione $h \in L^1(X)$ a valori in \mathbb{R} .

Esistono una varietà di metodi per il filtraggio delle immagini **basati sulle equazioni alle derivate parziali**. Le equazioni alle derivate parziali (PDE) in generale codificano l'evoluzione di un sistema. Per capire in che modo le PDE possono essere utilizzate per il filtraggio di immagini vediamo un importante risultato per la loro risoluzione.

Definizione 1.2.1. Si definisce soluzione fondamentale di un operatore differenziale $L \in \mathcal{D}(\mathbb{R}^n)$ una distribuzione u^* tale che, $Lu^* = \delta_0$.

Trovare la soluzione fondamentale di un operatore differenziale è molto importante nello studio delle equazioni alle derivate parziali siccome, trovata la soluzione fondamentale, siamo in grado di trovare qualsiasi soluzione particolare ci possa interessare, questo grazie a due particolari proprietà dell'operazione di convoluzione.

Proposizione 1.2.1. La prima di cui ci serviremo è: $\forall f \in \mathcal{D}(\mathbb{R}^n), f * \delta_0 = f$.

Tale proprietà può essere riassunta con la frase frase "la δ_0 è l'elemento neutro dell'operazione di convoluzione".

Proposizione 1.2.2. La seconda proprietà che ci interessa dice che

$$L(f * g) = L(f) * g = f * L(g).$$

Da queste due proprietà discende direttamente che, nota la soluzione fondamentale, la soluzione particolare del problema $Lu = f$ è $u = u^* * f$ con u^* soluzione fondamentale di L , infatti $Lu = L(u^* * f) = L(u^*) * f = \delta_0 * f = f$.

Questo vuol dire che nel caso $X = \mathbb{R}^2$ la soluzione fondamentale sarà una funzione definita $u^* : \mathbb{R}^2 \rightarrow \mathbb{R}$ e volendo applicare l'operatore differenziale L alla nostra immagine u_0 dovremmo fare la convoluzione $u^* * u_0$. Parlano in termini di quanto detto ad inizio paragrafo, applicare l'operatore differenziale vuol dire applicare un filtro e il filtro è proprio la soluzione fondamentale dell'operatore.

Come detto, le PDE codificano un'evoluzione, vedremo infatti che la soluzione di una PDE non dipende solo dai dati iniziali, ma anche da un altro parametro (tipicamente il tempo) che ci aiuterà a regolare in che misura vogliamo che il filtro sia applicato.

Il problema adesso è far eseguire questi calcoli ad un calcolatore, il quale non è in grado di lavorare con oggetti continui e richiede quindi di alcune approssimazioni per discretizzare il problema. A tal proposito la definizione di convoluzione può facilmente essere discretizzata parlando di successioni anziché di funzioni ed operando una sommatoria invece di un integrale.

$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m] g[n-m] = \sum_{m=-\infty}^{\infty} f[n-m] g[m].$$

Altri metodi ed approssimazioni saranno poi approfonditi durante la trattazione del problema.

1.3 Principali problemi di elaborazione digitale

Al giorno d'oggi i computer permettono una vasta gamma di elaborazioni digitali, ce ne sono alcuni, più semplici ma di notevole interesse, ormai già perfezionati ed altri che sono ancora oggetto di studio.

1.3.1 Rotazioni, riflessioni,etc

Le trasformazioni più semplici in assoluto riguardano movimenti rigidi come la **traslazione** o la **rotazione**, o anche omeomorfismi come la **riflessione**³. Questi sono molto utili per introdurre i primi concetti matematici, come l'impiego di matrici.

Definiamo una **matrice di trasformazione** come una matrice quadrata 3x3 tale che moltiplicata per un vettore di coordinate ci restituisca un nuovo vettore di coordinate. Applicare questo tipo di trasformazione vuol dire che il punto identificato dal primo sistema di coordinate va spostato nel punto identificato dal nuovo vettore di coordinate. Ad esempio:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

³Gonzalez and Wood [1992]

È una matrice di riflessione lungo l'asse verticale, infatti:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -x & y & 1 \end{bmatrix}$$

Il punto rimane cioè alla stessa ordinata ma si sposta lungo le ascisse da un valore di x ad uno di $-x$, che è esattamente ciò che si intende per riflessione lungo l'asse verticale. Eseguito questo calcolo per ogni punto appartenente all'immagine di partenza si ottiene il risultato desiderato. Altri esempi possono essere:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Riflessione lungo l'asse orizzontale

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scalamento

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotazione di un angolo theta

Questo tipo di trasformazioni sono solitamente dette **trasformazioni affini**

1.3.2 Cambio prospettiva

Una delle trasformazioni più semplici è quella del cambio prospettiva. Una volta compresa questa trasformazione si è fatto anche un primo passo per parlare di warping, che sarà trattato più avanti.

Un esempio pratico può essere: si ha la foto di un documento poggiato su una scrivania e la si vuole migliorare in modo da estrarne solo il documento e che i suoi bordi concidano quindi con i bordi dell'immagine.

In genere la cosa più semplice ed affidabile è quella di far scegliere i 4 angoli del documento ad un utente. Volendo automatizzare il processo si può però scegliere un'altra strada, ossia estrarre i bordi dell'immagine e cercare tra questi quelli che formano un trapezio; presumibilmente quello sarà il documento.

Ottenuti i 4 angoli, occorre calcolare la lunghezza e la larghezza della nuova immagine. Per fare ciò si può calcolare la lunghezza dei 4 lati banalmente con il teorema di Pitagora. A questo punto,

presi i lati a due a due non adiacenti, cioè che non hanno vertici in comune, ne si confrontano le lunghezze.

Per ogni coppia si adotta la lunghezza maggiore (si può anche usare quella minore ma è una scelta di scarso interesse per lo studio in atto).

Ottenuti questi dati si può calcolare la matrice di trasformazione. Essa sarà del tipo:

$\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix}$ dove $\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ definisce rotazione e scalamento, $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ è un vettore di traslazione e $[c_1 \ c_2]$ è un vettore di proiezione (che è nullo se il riquadro iniziale e quello finale coincidono).

L'intento è che, moltiplicando tale matrice per il vettore $\begin{bmatrix} \text{coordinata x} \\ \text{coordinata y} \\ 1 \end{bmatrix}$, si ottiene il vettore $\begin{bmatrix} \text{nuova coordinata } x^*k \\ \text{nuova coordinata } y^*k \\ k \end{bmatrix}$ eseguendo questa operazione per tutti i punti. Si costruisce l'immagine desiderata.

1.3.3 Morphing

Il morphing consiste nella trasformazione fluida, graduale tra due immagini di forma diversa, raffiguranti oggetti, persone, volti, paesaggi.

Il morphing è quindi una tecnica che combina l'uso in contemporanea di una dissolvenza incrociata e di un effetto di deformazione chiamato **warping** (termine inglese che significa appunto deformazione).

Per operare il warping si definiscono sull'immagine di partenza dei "punti chiave" che possono essere uniti tra di loro con delle linee operando così una "triangolarizzazione", si definiscono quindi sull'immagine di destinazione i corrispondenti punti e di conseguenza le corrispondenti linee.

Durante la dissolvenza dall'immagine iniziale a quella finale, le immagini vengono deformate tramite trasformazioni affini, facendo in modo che ciascun punto chiave si muova lungo il percorso che porta dalla sua posizione nell'immagine di partenza alla posizione del corrispondente punto nell'immagine di arrivo. Matematicamente, detto un punto chiave appartenente alla prima immagine p_0 e il corrispondente punto chiave nella seconda immagine p_1 , allora $\forall p_0$ appartenente alla prima immagine operiamo una traslazione del punto lungo il vettore differenza $v = p_1 - p_0$. Tutti i punti interni ai singoli triangoli verranno traslati di conseguenza. Questo è il processo di warping applicato alla prima immagine. Analogamente, applichiamo il processo di warping alla seconda immagine, cioè $\forall p_1$ appartenente alla seconda immagine, operiamo una

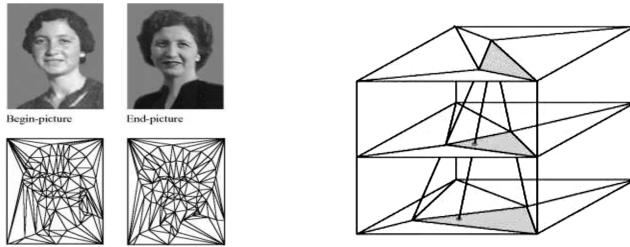


Figura 1.6. Definizione dei punti e delle linee sull’immagine di partenza e sull’immagine d’arrivo [Anton H. \[2010\]](#) e uno schema che mostra la trasformazione tra lo stato iniziale a quello finale di un singolo triangolo con un passaggio intermedio

traslazione del punto lungo il vettore differenza $-\mathbf{v} = p_0 - p_1$.

I passaggi intermedi saranno calcolati come segue: Sia $t \in (0,1)$, allora $p_t = p_0 vt = p_1(-v)t$. In questo modo, $\forall t \in (0,1)$ i punti chiave nelle due immagini si troveranno nella stessa posizione. Posta l’immagine iniziale e al di sopra di quella finale e applicata alla prima una dissolvenza di ottiene la trasformazione desiderata.



Figura 1.7. Processo di morphing con alcuni risultati intermedi.

Capitolo 2

Filtraggio digitale via Equazioni alle Derivate Parziali

2.1 Equazione del calore e la sua approssimazione numerica

L'equazione del calore, come facilmente intuibile, è stata formulata per determinare l'evoluzione di un sistema isolato che presenta al suo interno una data distribuzione di calore. La sua rappresentazione differenziale è la seguente:

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \Delta u(t, x) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}$$

Euristicamente, non è difficile pensare che si possano codificare (pensando ad un'immagine in bianco e nero) i pixel più luminosi come punti "*più caldi*", mentre quelli più scuri come punti "*più freddi*" ed applicare così l'equazione del calore all'immagine.

Mediante uno script MATLAB si può dunque osservare come quest'ultima operi nella pratica, sfruttando una approssimazione alle **differenze finite** utile per il calcolo delle derivate.

2.1.1 Metodo delle differenze finite

Il metodo delle differenze finite, come anticipato, viene impiegato nel calcolo approssimato delle derivate.

Definizione 2.1.1. *Data una generica funzione u , consideriamone lo sviluppo di Taylor*

$$u(x_i + \Delta(x)) = u(x_i) + u'(x_i)\Delta(x) + \frac{1}{2}u''(x_i)\Delta(x)^2 + o(h^2)$$

La scelta adottata per la suddetta approssimazione risulterà dunque essere:

$$\frac{du}{dx} \approx \frac{u(x+\Delta(x))-u(x)}{\Delta(x)} \approx \frac{u_{i+1}-u_i}{\Delta(x)}. \text{¹}$$

¹ [Recktenwald \[2004\]](#)

Tale approssimazione è detta **differenza finita in avanti**. Analogamente si trova, come approssimazione altrettanto valida la **differenza finita all'indietro**:

$$\frac{du}{dx} \approx \frac{u(x) - u(x - \Delta(x))}{\Delta(x)} \approx \frac{u_i - u_{i-1}}{\Delta(x)}.$$

Definizione 2.1.2. Consideriamo adesso gli sviluppi di Taylor che hanno portato alle formule di approssimazione alle differenze finite appena viste e sottraiamo membro a membro.

$$\begin{aligned} u(x_i + \Delta(x)) &= u(x_i) + u'(x_i)\Delta(x) + \frac{1}{2}u''(x_i)\Delta(x)^2 + \frac{1}{6}u'''(x_i)\Delta(x)^3 + o(\Delta(x)^3) \\ u(x_i - \Delta(x)) &= u(x_i) - u'(x_i)\Delta(x) + \frac{1}{2}u''(x_i)\Delta(x)^2 - \frac{1}{6}u'''(x_i)\Delta(x)^3 + o(\Delta(x)^3) \\ \downarrow \\ u(x_i + \Delta(x)) - u(x_i - \Delta(x)) &= +2u'(x_i)\Delta(x) + 2\frac{1}{6}u'''(x_i)\Delta(x)^3 + o(\Delta(x)^3) \end{aligned}$$

Questi conti inducono l'approssimazione:

$$\frac{du}{dx} \approx \frac{u(x + \Delta(x)) - u(x - \Delta(x))}{2\Delta(x)} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta(x)}$$

Tale approssimazione è detta **differenza finita centrata**.

Dal momento che la derivata seconda coincide con la derivata della derivata prima, è intuitivo dare la seguente definizione per la sua approssimazione numerica:

Definizione 2.1.3. Consideriamo la funzione u e l'approssimazione della sua derivata alle differenze finite $\frac{du}{dx} = \frac{u_{i+1} - u_i}{\Delta(x)}$, applicando ad esso il metodo delle differenze finite troviamo:

$$\begin{aligned} \frac{d^2u}{dx^2} &\approx \frac{\left(\frac{u_{i+1} - u_i}{\Delta(x)}\right)_{i+1} - \left(\frac{u_{i+1} - u_i}{\Delta(x)}\right)_i}{\Delta(x)} = \frac{\frac{(u_{i+1} - u_i)_{i+1}}{\Delta(x)} - \frac{(u_{i+1} - u_i)_i}{\Delta(x)}}{\Delta(x)} \\ &= \frac{\frac{(u_{i+2} - u_{i+1})}{\Delta(x)} - \frac{(u_{i+1} - u_i)}{\Delta(x)}}{\Delta(x)} = \frac{\frac{(u_{i+2} - u_{i+1}) - (u_{i+1} - u_i)}{\Delta(x)}}{\Delta(x)} \\ &= \frac{(u_{i+2} - u_{i+1}) - (u_{i+1} - u_i)}{\Delta(x)^2} = \frac{u_{i+2} - u_{i+1} - u_{i+1} + u_i}{\Delta(x)^2}. \end{aligned}$$

Osservazione 2.1.1. Per ottenere una stima accurata è bene utilizzare un valore di $\Delta(x)$ quanto più basso possibile. La migliore è guardare la differenza tra un pixel e quello adiacente ossia $\Delta(x) = 1$, ma allora

$$\frac{d^2u}{dx^2} \approx \frac{u_{i+2} - u_{i+1} - u_{i+1} + u_i}{\Delta(x)^2} = u_{i+2} - 2u_{i+1} + u_i.$$

È chiaro che scorrendo tutti gli indici questo è equivalente a $u_{i+1} - 2u_i + u_{i-1}$.

In sintesi si può utilizzare per il calcolo delle derivate seconde che compongono il laplaciano l'approssimazione

$$\frac{d^2u}{dx^2} \approx u_{i+1} - 2u_i + u_{i-1}$$

Errore di troncamento

Vale la pena di fare qualche considerazione sull'errore di troncamento che si commette adottando queste approssimazioni.

Definizione 2.1.4. Definiamo **errore** o più precisamente **errore assoluto** ϵ di un'approssimazione u' di un valore d'interesse u la differenza $\epsilon = u - u'$, cioè tra il valore esatto e quello approssimato.

Per quanto riguarda le derivate prime basta guardare agli sviluppi di Taylor che ci hanno portato alle loro approssimazioni per osservare che

Osservazione 2.1.2. Applicando la definizione di errore assoluto alle diverse approssimazioni finora definite per le derivate di primo grado troviamo:

- Differenza finita in avanti

$$u(x_i + \Delta(x)) = u(x_i) + u'(x_i)\Delta(x) + \frac{1}{2}u''(x_i)\Delta(x)^2 + o(\Delta(x)^2)$$

$$\downarrow$$

$$\epsilon = \frac{u(x_i + \Delta(x)) - u(x_i)}{\Delta(x)} - u'(x_i) \approx \frac{1}{2}u''(x_i)\Delta(x)$$

- Differenza finita all'indietro

$$u(x_i - \Delta(x)) = u(x_i) - u'(x_i)\Delta(x) + \frac{1}{2}u''(x_i)\Delta(x)^2 + o(\Delta(x)^2)$$

$$\downarrow$$

$$\epsilon = \frac{u(x_i) - u(x_i - \Delta(x))}{\Delta(x)} - u'(x_i) \approx \frac{1}{2}u''(x_i)\Delta(x)$$

- Differenza finita centrale

$$u(x_i + \Delta(x)) - u(x_i - \Delta(x)) = +2u'(x_i)\Delta(x) + 2\frac{1}{6}u'''(x_i)\Delta(x)^3 + o(\Delta(x)^3)$$

$$\downarrow$$

$$\epsilon = \frac{u(x_i + \Delta(x)) - u(x_i - \Delta(x))}{2\Delta(x)} - u'(x_i) \approx +\frac{1}{6}u'''(x_i)\Delta(x)^2$$

Ad un'attenta analisi possiamo vedere che tra i metodi di approssimazione in avanti, in indietro o centrale, il calcolo dell'errore portato dai primi due sono uguali tra di loro, quello centrale invece è diverso, esso dipende da un $\Delta(x)^2$ e non da un $\Delta(x)$, questo vuol dire che per $\Delta(x) < 1$ funziona meglio, per $\Delta(x) > 1$ funziona peggio. Nel caso in analisi $\Delta(x) = 1$ quindi la scelta è indifferente.

Per quanto riguarda la derivata seconda invece

Osservazione 2.1.3. Applicando la definizione di errore per la formula alle derivate finite per l'approssimazione della la derivata seconda:

$$\epsilon = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta(x)^2} - u_i''$$

Da questa rappresentazione non riusciamo purtroppo a dedurre nulla, proviamo quindi con un altro metodo. Consideriamo lo sviluppo di Taylor troncato al quarto ordine

$$u_{i+1} = u_i + u'_i\Delta(x) + \frac{1}{2}u''_i\Delta(x)^2 + \frac{1}{6}u'''_i\Delta(x)^3 + \frac{1}{24}u''''_i\Delta(x)^4 + o(\Delta(x)^4)$$

$$\begin{aligned}
 -\frac{1}{2}u_i''\Delta(x)^2 &= -u_{i+1} + u_i + u_i'\Delta(x) + \frac{1}{6}u_i'''\Delta(x)^3 + \frac{1}{24}u_i''''\Delta(x)^4 + o(\Delta(x)^4) \\
 -\frac{1}{2}u_i''\Delta(x)^2 &= -u_{i+1} + u_i + \left(\frac{u_{i+1} - u_{i-1}}{2\Delta(x)} - \frac{1}{6}u_i'''\Delta(x)^2\right)\Delta(x) + \frac{1}{6}u_i'''\Delta(x)^3 + \frac{1}{24}u_i''''\Delta(x)^4 + o(\Delta(x)^4) \\
 -\frac{1}{2}u_i''\Delta(x)^2 &= -u_{i+1} + u_i + \frac{1}{2}u_{i+1} - \frac{1}{2}u_{i-1} - \frac{1}{6}u_i''\Delta(x)^3 + \frac{1}{6}u_i'''\Delta(x)^3 + \frac{1}{24}u_i''''\Delta(x)^4 + o(\Delta(x)^4) \\
 -\frac{1}{2}u_i''\Delta(x)^2 &= -\frac{1}{2}u_{i+1} + u_i - \frac{1}{2}u_{i-1} + \frac{1}{24}u_i''''\Delta(x)^4 + o(\Delta(x)^4) \\
 \epsilon &= -u_i'' + \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta(x)^2} \approx \frac{1}{12}u_i''''\Delta(x)^2
 \end{aligned}$$

Risulta quindi evidente che l'errore dipende da $\Delta(x)^2$.

Per brevità di notazione poniamo d'ora in poi $\Delta(x) = h$.

Proposizione 2.1.1. *Dato un problema differenziale $-u'' + \sigma u = f$ con condizioni al contorno di Dirichlet g_0 e g_1 , allora se $\sigma \geq 0$ la risoluzione del problema tramite metodo delle differenze finite è convergente.*

Dimostrazione. Senza perdere di generalità possiamo assumere che u, f, σ definite su $I = (0,1)$, il problema assume quindi la forma

$$\begin{cases} -u'' + \sigma u = f \\ u(0) = g_0 \\ u(1) = g_1 \end{cases}$$

Per quanto visto nell'osservazione 2.1.3 possiamo scrivere

$$u''(x_j) = \frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} - \frac{1}{12}u^{(4)}(\xi_j)h^2$$

dove ξ_j è un punto opportuno in (x_{j-1}, x_{j+1}) . Allora per $j = 1, \dots, N-1$, si può scrivere che

$$\frac{-u(x_{j+1}) + 2u(x_j) - u(x_{j-1})}{h^2} + \frac{1}{12}u^{(4)}(\xi_j)h^2 + \sigma(x_j)u(x_j) = f(x_j).$$

Introducendo la notazione $\tau_j = \frac{1}{12}u''''(\xi_j)h^2$ (errore di troncamento locale) e ponendo:

$$\begin{aligned}
 \mathbf{u} &= [u_x \dots u_{N-1}]^T \\
 \boldsymbol{\tau}_h &= [\tau_x \dots \tau_{N-1}]^T \\
 \mathbf{b}_h &= [(f(x_1) + \frac{g_0}{h^2}, f(x_2), \dots, f(x_{N-2}), f(x_{N-1}) + \frac{g_1}{h^2})]^T
 \end{aligned}$$

si può allora scrivere in forma matriciale, $A_h \mathbf{u} = \mathbf{b}_h + \boldsymbol{\tau}_h$ dove
 $A_h = \frac{1}{h^2}tridiag(-1, 2, -1) + diag(\sigma_1, \dots, \sigma_{N-1})$ e $\sigma_j = \sigma(x_j)$.

Il metodo alle differenze finite consiste allora nel determinare un'approssimazione u_h di u andando a risolvere il sistema lineare $A_h \mathbf{u}_h = \mathbf{b}_h$. Osserviamo che \mathbf{u}_h risulta ben definito in quanto A_h è una matrice non singolare, il che è verificato in quanto la matrice è composta da sole righe linearmente indipendenti.

Risultando che τ_h tende a zero quando h tende a zero, il metodo dicesi consistente. In particolare risulta $\tau_h = O(h^2)$.

Tuttavia la consistenza non assicura da sola la convergenza del metodo.

Per studiarne la convergenza è necessario considerare il comportamento dell'errore

$\epsilon_h = \mathbf{u}_h - \mathbf{u}$ quando h tende a zero. Dato che risulta $A_h \epsilon_h = \tau_h$, e quindi $\epsilon_h = A_h^{-1} \tau_h$ si può quindi scrivere: $\|\epsilon_h\| \geq \|A_h^{-1}\| \|\tau_h\|$

Il passaggio successivo è far vedere che, lavorando in norma infinito, si è in grado di trovare una costante che, per ogni h , maggiora $\|A_h^{-1}\|_\infty$.

A questo scopo osserviamo che si può dimostrare che sia A_h che la matrice

$A_{0h} = \frac{1}{h^2} \text{tridiag}(-1, 2, -1)$ hanno inversa non negativa, e si ha:

$$A_{0h}^{-1} - A_h^{-1} = A_{0h}^{-1}(A_h - A_{0h})A_h^{-1} \geq 0.$$

Per come è definita A_h , vale la diseguaglianza $\|A_h^{-1}\|_\infty \leq \|A_{0h}^{-1}\|_\infty$ si può quindi osservare che $\|A_{0h}^{-1}\|_\infty = \max_j (A_{0h}^{-1} \text{ones}(N-1))_j$ dove $\text{ones}(N-1)$ indica il vettore di tutti 1.

Osservando che la soluzione esatta del problema $-u'' = 1, u(0) = u(1) = 0$, è il polinomio di secondo grado $\phi(x) = \frac{x(1-x)}{2}$, si può concludere che $A_{0h}^{-1}(\text{ones}(N-1))_j = \phi(x_j)$ e quindi che $\|A_h^{-1}\|_\infty \leq \|A_{0h}^{-1}\|_\infty \leq \max_{0 < x < 1} |\phi_x|$.

Questo risultato di stabilità ci permette di concludere che l'errore ϵ_h ha lo stesso ordine dell'errore di truncamento τ_h e di conseguenza che il metodo è convergente del secondo ordine.

□

Osservazione 2.1.4. Si noti che l'uniforme limitatezza della norma di A_h^{-1} implica che il metodo numerico sia stabile.

Questo è un risultato che ha valenza generale, riportato al caso dell'equazione del calore abbiamo che $\frac{\partial u}{\partial t} - \Delta u = \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0$ per cui $-\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}|_x$ e $-\frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}|_y$. Per entrambe le equazioni trovate ci troviamo nel caso $\sigma = 0$ e quindi in particolare $\sigma \geq 0$, in queste condizioni $\|A_h^{-1}\|_\infty = \|A_{0h}^{-1}\|_\infty$ e quindi in particolare $\|A_h^{-1}\|_\infty \leq \|A_{0h}^{-1}\|_\infty$, la condizione di convergenza è dunque soddisfatta.

2.1.2 L'equazione del calore

Il metodo delle differenze finite sarà impiegato in uno script MATLAB per l'implementazione dell'equazione del calore, è bene quindi prenderla in esame.

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \Delta u(t, x) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}$$

Ricordiamo $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ per cui la sua approssimazione numerica sarà:

$$u_{i,j}^{n+1} = u_{i,j}^n + k_x \Delta t (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + k_y \Delta t (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

Con k_x e k_y coefficienti di controllo (nello script porremo $k_x = k_y = k$).

L'equazione del calore, come anticipato, determina l'evoluzione di un sistema isolato che presenta al suo interno una data distribuzione di calore. È banale pensare che con il passare del tempo il calore si distribuisca, tendendo per un tempo infinito ad una distribuzione uniforme.

Applicando l'equazione del calore si ottiene quindi un'immagine sempre più "*liscia*", di fatto una sfocatura, e per un numero di iterazioni idealmente infinito si giungerebbe ad una distribuzione uniforme di colore, ossia una tinta unita.

Il seguente script MATLAB illustra come questo processo opera nella pratica.

```

1 Im=imread('parrot.jpeg'); %Apro l'immagine
2 Im=rgb2gray(Im); %La trasformo in bianco e nero
3 Im=imnoise(Im, 'gaussian'); %Aggiungo del rumore
4
5 %---Definisco le costanti e le condizioni iniziali
6
7 [ny, nx, ~]=size(Im); %Dimensioni dell'immagine
8 dt=0.25; %Passo temporale
9 u=double(Im); %Copia dell'immagine originale su cui
                  %lavorare
10 T=3; %Tempo, ossia T/dt + 1 definisce il numero
          %di iterazioni da eseguire
11 k=0.5;
12
13 %---Mostro l'immagine originale
14 imshow(uint8(Im));
15 title('immagine originale');
16
17 %---Metodo eq del calore
18 u=double(Im);
19 for t = 0:dt:T
20     u_xx = u(:,[2:ny ny],:) - 2*u + u(:,[1 1:ny-1],:); % derivata
                                                               % seconda lungo x
21     u_yy = u([2:ny ny],:,:)- 2*u + u([1 1:ny-1],:,:); % derivata
                                                               % seconda lungo y
22     u= u + k*dt*(u_xx+u_yy);
23     temp=u;
24 end

```

Provando a cambiare il tempo, ossia il numero di iterazioni, si può osservare come un maggior lasso di tempo produca immagini più sfocate.

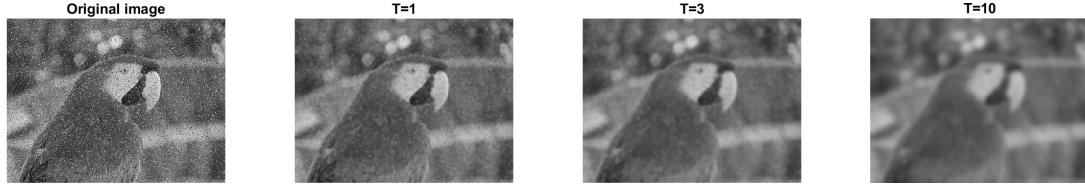


Figura 2.1. Effetti dell’equazione del calore nel tempo.

Questo metodo però ha un importante difetto, il rumore viene effettivamente eliminato o almeno ridotto ma si perdono importanti dettagli. In particolare, i margini degli oggetti presenti nell’immagine tendono progressivamente ad attenuarsi, come diretta conseguenza delle proprietà di regolarizzazione della soluzione dell’equazione del calore. Esistono tuttavia procedimenti come il metodo Perona-Malik che sono decisamente più efficaci.

2.1.3 Studio della stabilità del metodo

La tecnica delle differenze finite fornisce uno schema numerico che non è incondizionatamente stabile. Per studiare la sua stabilità e, conseguentemente, la sua convergenza, (grazie al teorema di equivalenza di Lax) utilizziamo un metodo classico per lo studio delle approssimazioni per le PDE: il metodo di Von Neumann.²

Il metodo di Von Neuman si basa sulla decomposizione degli errori in serie di Fourier
Prima di procedere con il calcolo è utile fare un’osservazione preliminare.

Osservazione 2.1.5. *Noto che il problema è risolubile tramite la formula iterativa*

$$u_{i,j}^{n+1} = u_{i,j}^n + r_x(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + r_y(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

essa può essere riscritta come

$$u_{i,j}^{n+1} = \frac{1}{2}u_{i,j}^n + r_x(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{1}{2}u_{i,j}r_y(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

risultando essere somma di due fattori speculari riguardanti le due coordinate, possiamo quindi ragionevolmente effettuare i calcoli per una delle due parti e i risultati saranno coerenti anche nell’altra parte.

Detto $\epsilon = N - u$ l’errore dove u è la soluzione calcolata dall’algoritmo in precisione infinita e N è la soluzione calcolata in precisione finita di calcolo, allora

$$u_j^{n+1} = \frac{1}{2}u_j^n + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n) \text{ e } N_j^{n+1} = \frac{1}{2}N_j^n + r(N_{j+1}^n - 2N_j^n + N_{j-1}^n) \text{ da cui}$$

$$\epsilon_j^{n+1} = N_j^{n+1} - u_j^{n+1} = \frac{1}{2}N_j - \frac{1}{2}u_j^n + r(N_{j+1} - u_{j+1}^n - 2N_j + 2u_j^n + N_{j-1} - u_{j-1}^n) = \frac{1}{2}\epsilon_j^n + r(\epsilon_{j+1}^n - 2\epsilon_j^n + \epsilon_{j-1}^n)$$

²Quarteroni [2010]

Sommiamo le due parti ritroviamo

$$\epsilon_{i,j}^{n+1} = \epsilon_{i,j}^n + r_x(\epsilon_{i+1,j}^n - 2\epsilon_{i,j}^n + \epsilon_{i-1,j}^n) + r_y(\epsilon_{i,j+1}^n - 2\epsilon_{i,j}^n + \epsilon_{i,j-1}^n)$$

questo dimostra che la soluzione e l'errore hanno lo stesso andamento.

Questa osservazione, oltre a fornirci un risultato necessario per il calcolo che ci accingiamo a fare, ci introduce al ragionamento per *separazione delle variabili* che utilizzeremo nella dimostrazione della proposizione seguente ed è molto utile per semplificare i calcoli.

Proposizione 2.1.2. *il problema*

$$u_{i,j}^{n+1} = u_{i,j}^n + r_x(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + r_y(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

dove $r_x = k_x \frac{\Delta t}{(\Delta x)^2}$ e $r_y = k_y \frac{\Delta t}{(\Delta y)^2}$, risulta stabile $\Leftrightarrow r_x + r_y \leq \frac{1}{2}$

Dimostrazione. Senza perdere di generalità, consideriamo il caso dell'equazione del calore **uno-dimensionale**, come visto essa può essere discretizzata come:

$$u_j^{n+1} = u_j^n + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

dove $r = k \frac{\Delta t}{(\Delta x)^2}$.

La variazione spaziale dell'errore può essere espansa con un integrale di Fourier rispetto ad x, cioè:

$$\epsilon(x, t) = \int_{\tilde{v}=-\frac{\pi}{\Delta x}}^{\tilde{v}=\frac{\pi}{\Delta x}} E_{\tilde{v}}(t) e^{i\tilde{v}x} d\tilde{v}$$

Essendo l'equazione lineare, il termine generico va come l'integrale stesso, ossia:

$$\epsilon_j^n = E_{\tilde{v}}(t) e^{i\tilde{v}_m x}$$
 da cui:

$$\epsilon_j^{n+1} = E_{\tilde{v}}(t + \Delta t) e^{i\tilde{v}x}$$

$$\epsilon_{j+1}^n = E_{\tilde{v}}(t) e^{i\tilde{v}(x+\Delta x)}$$

$$\epsilon_{j-1}^n = E_{\tilde{v}}(t) e^{i\tilde{v}(x-\Delta x)}$$

Sostituendo questi valori in $\epsilon_j^{n+1} = \epsilon_j^n + r(\epsilon_{j+1}^n - 2\epsilon_j^n + \epsilon_{j-1}^n)$ si ottiene:

$$E_{\tilde{v}}(t + \Delta t) e^{i\tilde{v}x} = E_{\tilde{v}}(t) e^{i\tilde{v}x} + r(E_{\tilde{v}}(t) e^{i\tilde{v}(x+\Delta x)} - 2E_{\tilde{v}}(t) e^{i\tilde{v}x} + E_{\tilde{v}}(t) e^{i\tilde{v}(x-\Delta x)})$$

Affinché il metodo sia stabile occorre che l'errore non aumenti mai, ossia che

$|E_{\tilde{v}}(t + \Delta t)| \leq |E_{\tilde{v}}(t)| \Rightarrow \left| \frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)} \right| \leq 1$. Esplicitiamo quindi per $\frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)}$ ed otteniamo:

$$\frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)} = 1 + r(e^{i\tilde{v}\Delta x} + e^{-i\tilde{v}\Delta x} - 2).$$

detto $\theta = \tilde{v}\Delta x$, ricordo $\tilde{v} \in [-\frac{\pi}{\Delta x}, \frac{\pi}{\Delta x}] \Rightarrow \theta \in [-\pi, \pi]$ per cui l'equazione diventa

$$\frac{E_{\tilde{v}}(t + \Delta t)}{E_m(t)} = 1 + r(e^{i\theta x} + e^{-i\theta x} - 2).$$

Prendiamo ora in considerazione l'identità

$$\sin\left(\frac{\theta}{2}\right) = \frac{e^{i\theta/2} - e^{-i\theta/2}}{2i} \Rightarrow \sin^2\left(\frac{\theta}{2}\right) = -\frac{e^{i\theta} + e^{-i\theta} - 2}{4}$$

alla luce di questa osservazione, l'equazione diventa:

$$\frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)} = 1 - 4r \sin^2\left(\frac{\theta}{2}\right).$$

Come detto, il metodo è stabile $\Leftrightarrow | \frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)} | \leq 1$ ma visto che $\frac{E_{\tilde{v}}(t + \Delta t)}{E_{\tilde{v}}(t)} = 1 - 4r \sin^2\left(\frac{\theta}{2}\right) \Rightarrow$ il metodo risulta stabile $\Leftrightarrow |1 - 4r \sin^2\left(\frac{\theta}{2}\right)| \leq 1$. Esplicitando:

$$-1 \leq 1 - 4r \sin^2\left(\frac{\theta}{2}\right) \leq 1 \Rightarrow -2 \leq -4r \sin^2\left(\frac{\theta}{2}\right) \leq 0 \Rightarrow 0 \leq 4r \sin^2\left(\frac{\theta}{2}\right) \leq 2$$

ma $\sin^2\left(\frac{\theta}{2}\right) \geq 0 \forall \theta \Rightarrow$ il metodo risulta stabile $\Leftrightarrow r \sin^2\left(\frac{\theta}{2}\right) \leq \frac{1}{2}$ siccome $\sin^2\left(\frac{\theta}{2}\right) \leq 1 \forall \theta \Rightarrow$ la condizione è sicuramente soddisfatta se $r \leq \frac{1}{2}$. Avendo definito $r = k \frac{\Delta t}{(\Delta x)^2} \Rightarrow$ condizione sufficiente affinché il metodo sia stabile è $k \frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$

Nel caso 2-dimensionale la soluzione discreta della PDE è:

$$u_{i,j}^{n+1} = u_{i,j}^n + r_x(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + r_y(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

dove $r_x = k_x \frac{\Delta t}{(\Delta x)^2}$ e $r_y = k_y \frac{\Delta t}{(\Delta y)^2}$

per cui la condizione diventa $r_x + r_y = k_x \frac{\Delta t}{(\Delta x)^2} + k_y \frac{\Delta t}{(\Delta y)^2} \leq \frac{1}{2}$.

□

Nel caso in esame $\Delta x = \Delta y = 1$ e $k_x = k_y \Rightarrow r_x + r_y = k \Delta t + k \Delta t \leq \frac{1}{2} \Rightarrow k \Delta t \leq \frac{1}{4}$.

La condizione appena calcolata è detta **condizione di Courant-Friedrichs-Lowy** (abbreviato **CFL**) per le equazioni alle derivate parziali di tipo parabolico

2.2 Metodo Perona-Malik

Il metodo Perona-Malik, come anticipato, si basa sull'equazione del calore. L'idea è quella di applicare tale equazione lontano dai bordi dell'immagine e mantenere invece inalterati quest'ultimi. Questo metodo risulta particolarmente efficace per risolvere problemi di disturbo come ad esempio un rumore del tipo salt and pepper, cioè con dei pixel bianchi o neri sparsi per la foto.

È bene capire da un punto di vista prettamente matematico cosa vuol dire che il metodo si basa sull'equazione del calore, che ricordiamo essere:

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \Delta u(t, x) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}$$

Il laplaciano è la divergenza del gradiente, ed è qui che viene operata la modifica.

Come detto: $\Delta(u) = \operatorname{div}(\nabla(u))$, volendo operare un controllo si introducirà un coefficiente k ottenendo $\operatorname{div}(k\nabla(u))$, nel caso esso sia uno scalare costante, avremo

$\operatorname{div}(k\nabla(u)) = k\operatorname{div}(\nabla(u)) = k\Delta(u)$ ed è il caso implementato nelle pagine precedenti. Il metodo Perona-Malik invece prevede l'impiego di un coefficiente k non costante, ma che cambi a seconda del pixel su cui si opera, otteniamo così una matrice delle stesse dimensioni della matrice che rappresenta l'immagine e che funga da mappa, indicante per ogni pixel l'intensità con cui la diffusione va applicata. In questo senso, questo filtro viene detto di diffusione anisotropa. Il termine diffusione anisotropa indica, a livello globale, quella particolare proprietà secondo cui la diffusione del colore sull'immagine dipende fortemente dalla direzione presa in considerazione. Tuttavia è possibile constatare come questa dicitura sia in realtà un abuso di notazione di cui ci serviamo in termini prettamente esplicativi dal momento che il metodo in esame è localmente isotropo, si articola cioè lungo un'unica direzione variandone però l'intensità punto per punto. Esistono tuttavia dei metodi di diffusione anisotropa, in tal caso k sarà un tensore variabile.

Ritornando al metodo in esame, il problema affrontato diventa

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \operatorname{div}(k\nabla(u)) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ \frac{\partial u}{\partial N} = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}$$

Con k dipendente dal gradiente, in particolare $k = k(|\nabla(u)|^2)$.

Servirà dunque esplicitare e discretizzare questo problema, per fare ciò saranno impiegati, oltre ai metodi già visti, altri accorgimenti.

2.2.1 Soluzione discreta della PDE

Sia u una funzione, sia $D_f(x_i)$ la differenza finita in avanti, $D_b(x_i)$ la differenza finita all'indietro e sia $D_c(x_i)$ la differenza finita centrata, si può osservare che:

$$D_f(x_i) + D_b(x_i) = \frac{u_{i+1} - u_i}{\Delta(x)} + \frac{u_i - u_{i-1}}{\Delta(x)} = \frac{u_{i+1} - u_{i-1}}{\Delta(x)} = 2D_c(x_i)$$

Capiamo quindi che la differenza finita centrata è $D_c(x_i) = \frac{1}{2}(D_f(x_i) + D_b(x_i))$ cioè la media delle altre due.

Preso un punto si può operare le differenze finite nelle 4 direzioni, le indicheremo con N (Nord), S (Sud), W (Ovest) e E (Est). Notiamo allora che la differenza finita Nord è la differenza finita in avanti rispetto a y , mentre la differenza finita S è la differenza finita all'indietro rispetto a y , la loro media sarà quindi la differenza finita centrata rispetto a y . Analogamente la media delle differenze finite Ovest ed Est sarà la differenza finita centrata rispetto a x . In formule:

$$\frac{\partial u}{\partial x} \approx \frac{u_E + u_W}{2} \quad ; \quad \frac{\partial u}{\partial y} \approx \frac{u_N + u_S}{2}$$

Ricordiamo che il gradiente è il vettore delle derivate parziali, la divergenza invece è la somma delle derivate parziali. Calcoliamo quindi un'approssimazione discreta della PDE

$$\begin{aligned} \frac{\partial u}{\partial t}(t, x) &= \operatorname{div}(k \nabla(u)) = \\ &= \frac{k \nabla u}{\partial x} + \frac{k \nabla u}{\partial y} \\ &\approx \frac{k_N \nabla_N + k_S \nabla_S}{2} + \frac{k_W \nabla_W + k_E \nabla_E}{2} \\ &= \frac{1}{2}(k_N \nabla_N + k_S \nabla_S + k_W \nabla_W + k_E \nabla_E) \end{aligned}$$

Ci si ritrova quindi a risolvere l'equazione

$$\frac{\partial u}{\partial t}(t, x) = \frac{1}{2}(k_N \nabla_N + k_S \nabla_S + k_W \nabla_W + k_E \nabla_E)$$

che è della forma $y'(x) = f(x, y(x))$ e può quindi essere risolta con un metodo semplice. Ancora una volta ricorriamo alle differenze finite.

$$y'(x_i) \approx \frac{y(x_{i+1}) - y(x_i)}{\Delta t}$$

ma $y'(x) = f(x, y(x))$ questo vuol dire che

$$\frac{y(x_{i+1}) - y(x_i)}{\Delta t} \approx f(x, y(x)) \Rightarrow y_{i+1} \approx y_i + \Delta t f(x, y(x))$$

tale metodo è detto **metodo di Eulero esplicito**³

³Monegato [1998]

La soluzione discreta della PDE, con il metodo di Eulero esplicito, sarà quindi:

$$u_{i+1} = u_i + dt \frac{1}{2}(k_N \nabla_N + k_S \nabla_S + k_W \nabla_W + k_E \nabla_E)$$

Osservazione 2.2.1. Tramite il metodo di Von Neuman, in modo simile a quanto fatto per l'equazione del calore si dimostra che il metodo Perona-Malik risulta essere stabile $\iff 4 \frac{1}{2} \frac{dt}{\Delta x^2} \leq \frac{1}{2}$.

Ricordiamo che nel caso in esame, ove $\Delta x = 1$, la condizione diventa:

$$dt \leq \frac{1}{4}.$$

2.2.2 Maschere di convoluzione

Per implementare il filtro di diffusione anisotropica detto *Perona-Malik*, ci serviremo di alcune **maschere**.

Un filtro è una funzione $F : \mathbb{R}^2 \rightarrow \mathbb{R}$, in particolare è un polinomio a due incognite che, date in input delle coordinate restituirà un valore numerico, chiameremo tale coefficiente **peso**. Compiliamo quindi una maschera con i pesi calcolati in tal modo per ogni posizione ottenendo una cosa del tipo:

a_1	a_2	a_3
a_4	a_5	a_6
a_7	a_8	a_9

Figura 2.2. Esempio di maschera di convoluzione 3x3

Le dimensioni della maschera possono variare a discrezione del programmatore ma, solitamente si utilizzano maschere dimensioni piccole e dispari. Dispari perchè è importante individuare il centro della maschera. Piccole perchè l'utilizzo delle maschere porta degli effetti bordo che è bene limitare. Queste motivazioni verranno meglio spiegate a breve.

Come detto nelle nozioni introduttive, applicare un filtro vuol dire operare una convoluzione tra due funzioni: l'immagine ed il filtro stesso.

Le maschere sono un utile strumento proprio per il calcolo delle convoluzioni, motivo per cui sono solitamente dette **maschere di convoluzione**.

Per operare una convoluzione, detti $u : \mathbb{R}^2 \rightarrow \mathbb{R}$ l'immagine e $F : \mathbb{R}^2 \rightarrow \mathbb{R}$ il filtro, per definizione, $\forall p_{i,j} \in \text{dom}(u)$ si prende un intorno $I(p)$, le cui dimensioni dipendono da quelle

della maschera scelta per F . Successivamente si sommano i prodotti tra i valori di u e quelli di F , cioè i pesi della maschera, per ottenere in fine l'immagine filtrata.

In pratica, si scorre la maschera sui vari pixel dell'immagine

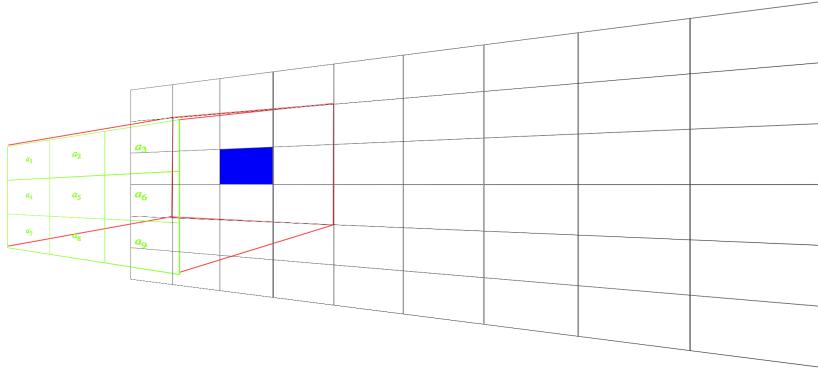


Figura 2.3. Griglia dell'immagine in nero, maschera in verde, pixel in esame in blu

Per ogni pixel ne viene ricalcolato il valore come segue

$$\begin{array}{|c|c|c|} \hline p_{i-1,j+1} & p_{i,j+1} & p_{i+1,j+1} \\ \hline p_{i-1,j} & p_{i,j} & p_{i+1,j} \\ \hline p_{i-1,j-1} & p_{i,j-1} & p_{i+1,j-1} \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline a_1 & a_2 & a_3 \\ \hline a_4 & a_5 & a_6 \\ \hline a_7 & a_8 & a_9 \\ \hline \end{array}$$

$$u(p_{i,j}) * F = a_1 p_{i-1,j+1} + a_2 p_{i,j+1} + a_3 p_{i+1,j+1} + a_4 p_{i-1,j} + a_5 p_{i,j} + a_6 p_{i+1,j} + a_7 p_{i-1,j-1} + a_8 p_{i,j-1} + a_9 p_{i+1,j-1}.$$

Figura 2.4. Intorno di un punto, maschera di convoluzione e soluzione analitica

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Ma allora una convoluzione con $a =$

vuol dire fare la derivata prima

approssimata alle differenze finite lungo y , in particolare verso l'alto, infatti facendo il conto di cui sopra si ottiene $u(p_{i,j}) * F = p_{i,j+1} - p_{i,j}$ che è esattamente quanto trovato analizzando il metodo alle differenze finite si può operare analogamente in tutte e 4 le direzioni.

Per facilitare la successiva implementazione del filtro, si è scritta una funzione MATLAB che operi in tal senso.

```

1 function B=convoluzione(A,k);
2
3 [r c] = size(A); %Memorizzo le dimensioni dell'immagine
4 [m n] = size(k); %Memorizzo le dimensioni della maschera
5 h = rot90(k, 2);
6
7 %Definisco una cornice per gestire gli effetti di bordo
8 center = floor((size(h)+1)/2);
9 left = center(2) - 1;
10 right = n - center(2);
11 top = center(1) - 1;
12 bottom = m - center(1);
13
14 %Preparo un "piano di lavoro"
15 Rep = zeros(r + top + bottom, c + left + right);
16 for x = 1 + top : r + top
17     for y = 1 + left : c + left
18         Rep(x,y) = A(x - top, y - left);
19     end
20 end
21
22 %Opero la convoluzione
23 B = zeros(r , c);
24 for x = 1 : r
25     for y = 1 : c
26         for i = 1 : m
27             for j = 1 : n
28                 q = x - 1;
29                 w = y - 1;
30                 B(x, y) = B(x, y) + (Rep(i + q, j + w) * h(i, j));
31             end
32         end
33     end
34 end

```

Notare che Rep ha dimensioni maggiori rispetto all'immagine questo perchè dobbiamo gestire anche i pixel sui bordi del riquadro dell'immagine. Volendo fare un esempio, si consideri il pixel in posizione (1,1), ossia l'angolo in alto a sinistra, allora i coeff a_1, a_2, a_3, a_4 e a_7 , non avranno nessun corrispettivo da moltiplicare, costruiamo quindi una cornice nera (cioè pixel di valore nullo) per ovviare a questo problema. Ovviamente se la maschera è grande, questo porterà a dei visibili errori di bordo, il che è esattamente il motivo per cui si usano solitamente maschere di dimensioni molto piccole.

2.2.3 Rilevamento dei bordi

Il metodo Perona-Malik serve ad eliminare il rumore, preservando i bordi. Per essere in grado di preservarli però dobbiamo prima essere in grado di riconoscerli. Il metodo prevede l'introduzione del termine $K(\Delta(u))$ che dipende quindi dal laplaciano dell'immagine che si intende filtrare. Cerchiamo di capire il perché di questa scelta.

Operando una derivata in una data direzione, per il significato in sè di derivata, questa assume valori più elevati quando la variazione è elevata, e assume valori nulli quando non c'è variazione in quella direzione. Per questo motivo, applicata ad un' immagine, ne rileviamo i bordi.

Pres a tinta unita la derivata sarà quindi nulla in ogni suo punto (è intuitivo: se un'immagine è una funzione che, date due coordinate restituisce un colore, allora una tinta unita è una funzione costante ed in quanto tale ha derivata nulla).

Operando una derivata seconda in una data direzione, per il significato in sè di derivata seconda, questa assume valori più elevati quando la concavità è più stretta, e assume valori pressocchè nulli quando non ci sono concavità (si può pensare alle concavità come a dei picchi o dei ventri, su di una immagine vuol dire chiazze di colore diverso).

Pres a sfumatura di colore che varia in maniera lineare, la derivata seconda sarà nulla in ogni suo punto, la derivata prima sarà invece costante.

x Per verificare la validità di questi concetti teorici si è implementato un semplice script

MATLAB che, presa un'immagine, opera il calcolo delle derivate e stampa a video le sole derivate sotto forma di immagine. Ne risulterà quindi un'immagine in bianco e nero con pixel quanto più chiari quanto più e alto il valore della derivata. Saranno mostrate come immagini diverse le derivate parziali, il gradiente ed il laplaciano dell'immagine data in input, così da poterli confrontare ed evidenziarne le differenze. Ci aspettiamo che le immagini risultanti corrispondano con i bordi dell'immagine originale.

Saranno mostrati i risultati prodotti da tale script dandogli in input immagini di diverso tipo, saranno poi commentati con osservazioni e considerazioni di vario tipo.

Si riporta lo script MATLAB appena citato.

```

1 %---Operazioni preliminari
2 Im=imread("nome_immagine.png"); %Apro l'immagine
3
4 [ny, nx, ~]=size(Im); %Memorizzo le dimensioni dell'immagine
5 u=double(Im); %Copia dell'immagine originale su cui
    lavorare
6 h=80; %Definisco un parametro che usero' per
        enfatizzare i bordi in fase di stampa
7
8
9 %---Calcolo tutte le derivate
10 u_x = u(:,[1 1:nx-1],:)-u; %derivata
    prima lungo x
11 u_xx = u(:,[2:nx nx],:)-2*u+u(:,[1 1:nx-1],:); %derivata
    seconda lungo x
12 u_y = u([1 1:ny-1],:,:)-u; %derivata
    prima lungo y
13 u_yy = u([2:ny ny],:,:)-2*u+u([1 1:ny-1],:,:); %derivata
    seconda lungo y
14 u_xy = u_x([1 1:ny-1],:,:)-u_x; %derivata
    seconda mista
15
16 %---Stampo i risultati
17 figure()
18 subplot(2,3,2),text(0.3,0,nome,'FontSize',20); axis off
19 subplot(2,3,4), imshow(Im)
20 title('Immagine originale')
21 subplot(2,3,5), imshow(uint8(h*abs(u_x)))
22 title('h*u_x')
23 subplot(2,3,6), imshow(uint8(h*abs(u_y)))
24 title('h*u_y')
25
26 figure()
27 subplot(2,3,2),text(0.3,0,nome,'FontSize',20); axis off
28 subplot(2,3,4), imshow(Im)
29 title('Immagine originale')
30 subplot(2,3,5), imshow(uint8(h*abs(u_x + u_y)))
31 title('h*(u_x + u_y)')
32 subplot(2,3,6), imshow(uint8(h*abs(u_xx + u_yy)))
33 title('h*(u_{xx} + u_{yy})')

```

Eseguiamo adesso questo script che ci permetterà di osservare, fornendogli in input diverse immagini molto semplici, se abbiamo ottenuto i risultati attesi.

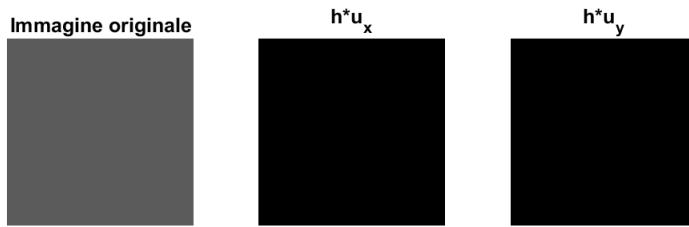
Esempio 1 - Tinta unita

Figura 2.5. Derivate parziali di una tinta unita.

Si può vedere come con un’immagine a tinta unita le derivate sono nulle, quindi lo saranno anche gradiente e laplaciano.

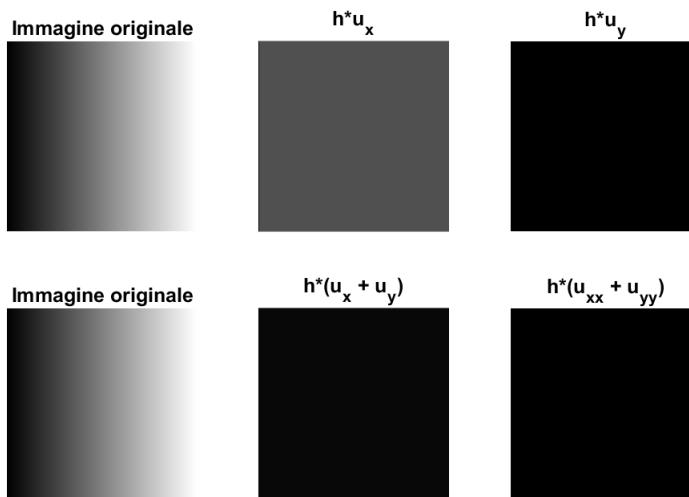
Esempio 2 - Sfumatura lungo l’asse x

Figura 2.6. Derivate parziali, gradiente e laplaciano di una sfumatura orizzontale.

Guardando invece ad una immagine che presenta una sfumatura lineare lungo l’asse x, la derivata lungo x assume un valore costante mentre la derivata lungo y è nulla, proprio perchè lungo y non c’è variazione mentre lungo x c’è una variazione costante.

Ovviamente, date queste premesse, il gradiente sarà costante uguale ad u_x (siccome $u_y = 0$) e quindi il laplaciano sarà nullo. Il fatto che in entrambi questi esempi il laplaciano sia nullo è un buon segno, lo useremo per rilevare i bordi ed in queste immagini non ve ne sono, quindi è giusto che il laplaciano sia nullo.

Esempio 3 - Parziale sfumatura lungo la diagonale

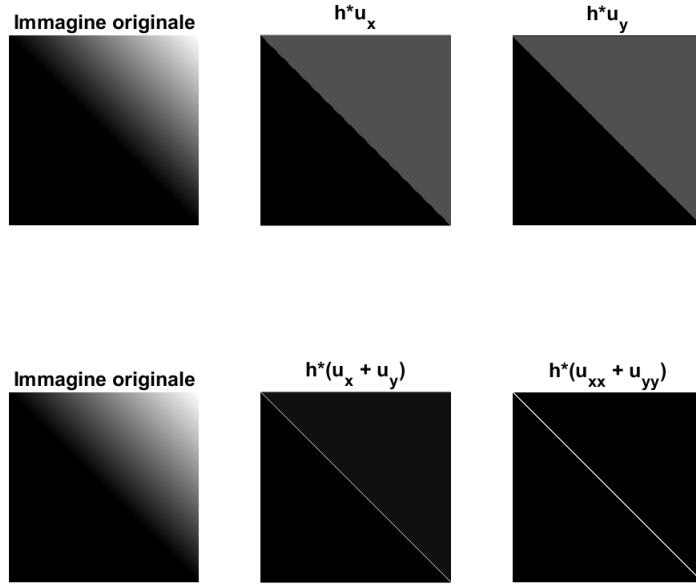


Figura 2.7. Derivate parziali, gradiente e laplaciano di una parziale sfumatura diagonale.

Presa una sfumatura diagonale, ma solo su metà immagine vediamo dei risultati interessanti: entrambe le derivate parziali sono nulle nelle regioni in cui non c’è sfumatura, esattamente come nel caso della tinta unita, ed entrambe sono costanti dove c’è sfumatura (che ricordiamo essere lineare).

Tutto ciò riconferma quanto visto dai punti precedenti, volgendo quindi uno sguardo al gradiente ed al laplaciano si può notare che mentre il gradiente ha un aspetto molto simile alle due derivate parziali, sommando i loro valori è semplicemente più luminoso, per quanto riguarda il laplaciano la storia cambia. Le derivate seconde sono indicative della variazione delle derivate prime, cioè della variazione della variazione del valore della funzione, ma l’unica variazione che hanno le derivate prime è lungo la diagonale. Abbiamo così individuato il nostro primo bordo, cioè la diagonale che divide di fatto due regioni, una in cui il colore è costante ed una in cui sfuma.

Si riportano ancora due varianti di un’ultima immagine esempio, provando ad introdurre una semplice figura.

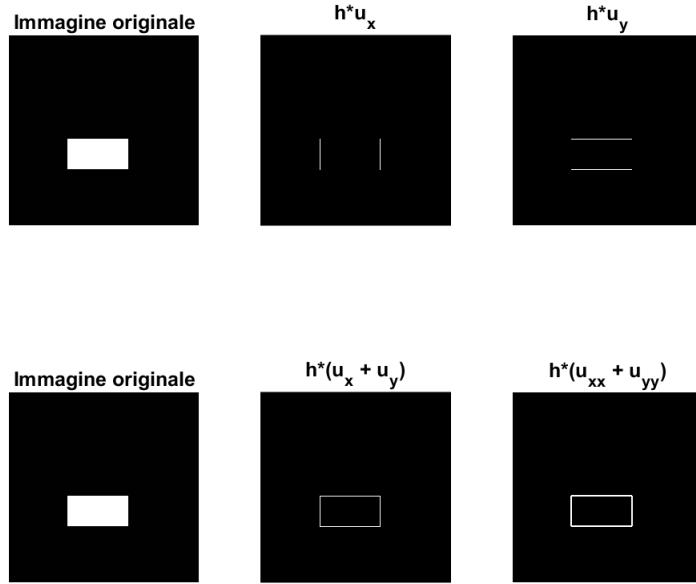
Esempio 4 - Rettangolo bianco su sfondo nero

Figura 2.8. Derivate parziali, gradiente e laplaciano di una figura semplice.

Importando un rettangolo bianco su di uno sfondo nero, come confermato anche dalla prima sfumatura, la derivata lungo x rileva i bordi verticali, quella lungo y i bordi orizzontali, dalla loro somma (quindi dal gradiente) otteniamo già il bordo del rettangolo. Il bordo così ottenuto è un bordo che idealmente rimarrà inalterato a prescindere dall'ordine della derivata, in particolare quindi anche per derivate seconde, quindi il laplaciano continua a soddisfare la richiesta di determinare i bordi.

Come detto: "*Il bordo così ottenuto è un bordo che idealmente rimarrà inalterato a prescindere dall'ordine della derivata*" è interessante capire perché. Presa una striscia di pixel, cioè uno strato dell'immagine (la si può immaginare quindi come una funzione $u_y : \mathbb{R} \rightarrow \mathbb{R}$), risulterà essa raffigurare una funzione porta!

La funzione porta non è derivabile in senso classico, ripensando alla definizione di derivata avremmo un valore di +infinito prima e -infinito poi. La sua derivata sarà quindi una coppia di delta di Dirac.

Proviamo in fine ad introdurre del rumore in quest'ultima immagine e osserviamo cosa accade.

Esempio 5 - Rettangolo bianco su sfondo dietro con rumore

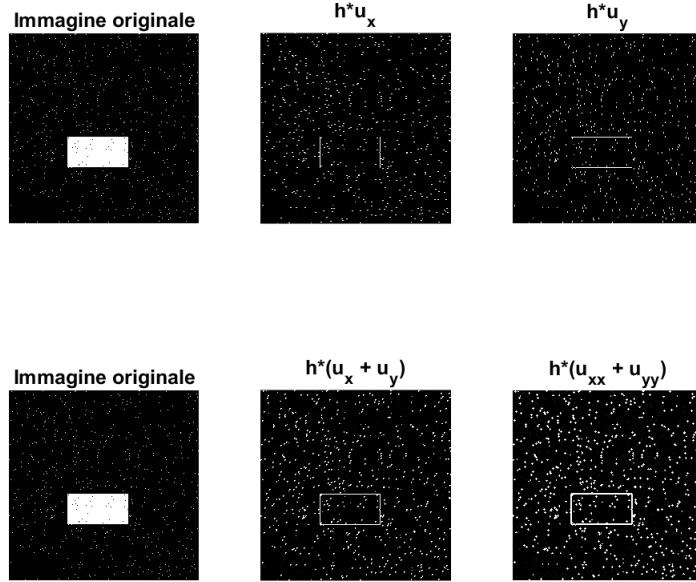


Figura 2.9. Derivate parziali, gradiente e laplaciano di una figura semplice con rumore.

Si può fare una considerazione: quando si trova un elemento di disturbo, come un puntino o una chiazza di colore, questo risulterà essere una forte variazione di colore improvvisa ed in quanto tale la derivata calcolata in un intorno di quel punto sarà, in valore assoluto, molto alta. Questo vuol dire che il gradiente ne rileverà i bordi e il metodo Perona-Malik non lo toccherà, questa cosa non va bene: è un elemento di disturbo e va quindi eliminato.

Una possibile soluzione a questo problema verrà illustrata in seguito.

2.2.4 Problema dei "bordi" d'interferenza

Come evidenziato con l'ultima immagine di prova nella sezione dedicata al rilevamento dei bordi, anche gli elementi di disturbo ne hanno e vengono quindi rilevati. Questo è un problema perché il metodo Perona-Malik tenderà a preservarli.

Per ovviare a questo problema operiamo su di una copia dell'immagine una diffusione come quella operata dall'equazione del calore vista in precedenza, così facendo in questa copia, i bordi dell'immagine risulteranno rovinati, ma non scomparsi! Gli elementi di disturbo saranno invece eliminati.

Moltiplicando i due gradienti così trovati si può osservare che: in corrispondenza degli elementi di disturbo il secondo gradiente sarà nullo ed il prodotto sarà dunque anch'esso nullo, vicino ai bordi il secondo gradiente non è nullo, ma il primo sì! Quindi il prodotto sarà ancora nullo, in corrispondenza degli effettivi bordi entrambi i gradienti saranno non nulli e quindi neanche il loro prodotto lo sarà. Questo vuol dire che:

- gli elementi di disturbo sono stati eliminati
- l'effetto distruttivo sui bordi non viene riportato
- gli effettivi bordi dell'immagine vengono preservati

I bordi sono quindi calcolati in maniera efficiente.

La funzione di controllo

È doveroso fare ancora una considerazione. Serve applicare ancora una trasformazione ai nostri bordi in modo da risultare ancor più efficienti ai nostri scopi, ossia una funzione di controllo. Cerchiamo una funzione decrescente tale che $k(0) = 1$ e $\lim_{s \rightarrow \infty} k(s) = 0$. Daremo in input a tale funzione il valore del gradiente calcolato in ogni pixel, otterremo così una mappatura che ci dice per ogni pixel in che misura operare la diffusione. Calato nel caso di studio, $k(0) = 1$ vuol dire che dove non c'è bordo opera l'equazione del calore mentre $\lim_{|\nabla(u)|^2 \rightarrow \infty} k(|\nabla(u)|^2) = 0$ vuol dire che tanto più i bordi sono accentuati, tanto più l'equazione del calore non deve operare. Facciamo questo banalmente perché considerando semplicemente i bordi, avremmo l'effetto opposto! Infatti dove non ci sono bordi $\nabla(u) = 0$ e quindi non ci sarebbe diffusione, dove ci sono bordi molto marcati $\nabla(u) \rightarrow \infty$ quindi la diffusione verrebbe adoperata addirittura più del normale! Non a caso una delle scelte più semplici, anche se non delle più classiche né efficienti è $k = \frac{1}{1+|\nabla(u)|^2/c}$ dove c è un fattore di controllo. Scelte decisamente più classiche ed impiegate sono:

- $k = e^{-\frac{|\nabla(u)|^2}{c}}$ che preserva maggiamente i bordi ad alto contrasto e meno quelli a basso contrasto
- $k = \frac{1}{\sqrt{1+(|\nabla(u)|^2/c)}}$ che preserva maggiormente regioni grandi piuttosto che quelle piccole

Come visto nella sezione 2.2.1 $\operatorname{div}(k\nabla(u)) = \frac{1}{2}(k_N \nabla_N + k_S \nabla_S k_W \nabla_W + k_E \nabla_E)$ calcoliamo quindi i vettori k_N , k_S , k_W e k_E dando in input alla funzione di controllo scelta i vettori ∇_N , ∇_S , ∇_W e ∇_E rispettivamente.

2.2.5 Implementazione

Nel corso della trattazione sono stati forniti tutti gli elementi che portano alla formazione di questo script. Ci si limiterà quindi a dei richiami e a delle precisazioni di tipo tecnico.

Prima di iniziare il filtraggio ci sono alcune operazioni preliminari da fare. In primo luogo occorrerà, ovviamente, caricare l'immagine che si intende filtrare. Successivamente l'immagine viene convertita in bianco e nero e viene aggiunto del rumore.

```
1 img=imread('nome_file.png'); %Apertura dell'immagine
2 img=rgb2gray(img); %Trasformazione in bianco e nero
3 im = imnoise(im,'salt & pepper',0.02); %Aggiunta del rumore
```

È importante precisare che quest'ultima operazione, in un caso reale, non ha senso di esistere ma è messa lì per il puro scopo di simulare il problema che intendiamo risolvere.

```
4 % conversione in double per il calcolo.
5 im = double(im);
```

Di norma codifichiamo le immagini come uint8 (interi senza segno da 0 a 255), tuttavia mantenere questa formattazione durante il calcolo potrebbe portare ad errori di approssimazione numerica assolutamente non trascurabili. Consideriamo quindi matrici a valori reali, approssimate dalla macchina a numeri macchina a doppia precisione.

```
6 % Condizioni iniziali della PDE.
7 diff_im = im;
8
9 num_iter=20; %numero di iterazioni
10 delta_t=0.1; %costante d'integrazione
11 c=60; %coefficiente di controllo del gradiente
12
13 sigma=1; %costante di controllo della diffusione uniforme
```

Occorre settare alcuni parametri per il calcolo:

- Numero di iterazioni per il metodo di Eulero: maggiore è il valore di questo parametro e più volte verrà applicato il metodo
 - Costante d'integrazione: maggiore è il valore di questo parametro e maggiore sarà l'intensità con cui viene applicato il metodo
- N.B. Sia aumentando il primo parametro sia il secondo ciò che otterremo sarà un'immagine filtrata maggiormente. La differenza è che aumentando la costante di integrazione applicheremo il metodo con maggior intensità ad ogni passo; invece aumentando il numero di iterazioni applicheremo il metodo un numero maggiore di volte. Ad ogni iterazione tutti i calcoli andranno rifatti, ciò vuol dire che aumentando il numero di iterazioni e riducendo la costante di integrazione, avremo un'immagine filtrata più finemente a discapito di un grosso aumento di costo computazionale*
- Coefficiente di controllo del gradiente: permette di far risaltare di più (se $c < 1$) o di meno (se $c > 1$) i bordi

- costante di controllo diffusione uniforme: il calcolo del vettore c utilizzato per modificare l'equazione del calore avviene tramite gradiente. Come già analizzato nella sezione 5.3, conviene fare questo calcolo su una versione sfocata dell'immagine, per fare ciò applichiamo l'equazione del calore. Modificare questo coefficiente significa modificare l'intensità con cui viene applicata. Un valore più alto sfoca di più e porta ad eliminare più rapidamente il rumore a discapito della definizione dei bordi.

Settati i parametri inizia il calcolo effettivo

```

15 % Maschera di convoluzione
16 hN = [0 1 0; 0 -1 0; 0 0 0];
17 hS = [0 0 0; 0 -1 0; 0 1 0];
18 hE = [0 0 0; 0 -1 1; 0 0 0];
19 hW = [0 0 0; 1 -1 0; 0 0 0];
20
21
22
23 for t = 1:num_iter
24
25 % Calcolo alle differenze finite nelle 4 direzioni
26 nablaN = convoluzione(diff_im,hN);
27 nablaS = convoluzione(diff_im,hS);
28 nablaW = convoluzione(diff_im,hW);
29 nablaE = convoluzione(diff_im,hE);
30
31 diff_blur = f_eq_del_calore(diff_im,delta_t,num_iter,sigma);
32 nablaN_blur = convoluzione(diff_blur,hN);
33 nablaS_blur = convoluzione(diff_blur,hS);
34 nablaW_blur = convoluzione(diff_blur,hW);
35 nablaE_blur = convoluzione(diff_blur,hE);
36
37
38 kN = exp(-(nablaN_blur/c).^2);
39 kS = exp(-(nablaS_blur/c).^2);
40 kW = exp(-(nablaW_blur/c).^2);
41 kE = exp(-(nablaE_blur/c).^2);

```

Calcolati tutti i fattori possiamo in fine applicare il metodo utilizzando la formula trovata nella sezione 5.1

```

42 % Soluzione discreta della PDE.
43 diff_im = diff_im + delta_t*(kN.*nablaN + kS.*nablaS + kW.*nablaW +
kE.*nablaE );

```

Concludiamo stampando l'immagine rumorosa e quella filtrata per poter apprezzare gli effetti sortiti

```

45 % Stampa di controllo
46 fprintf('\rIteration %d\n',t);
47 end
48 % Stampa dei risultati
49 figure('Name','Original picture'); %Stampa dell'immagine rumorosa
50 imshow(im);
51 figure('Name','Perona Malik'); %Stampa dell'immagine filtrata
52 imshow(uint8(diff_im));

```


Capitolo 3

Esempi di utilizzo

3.1 Esempi di utilizzo

Ultimata l'implementazione del metodo è bene spendere del tempo per testarne l'efficacia ed osservare come i vari parametri influenzano i risultati ottenuti.

Richiamiamo il significato teorico dei vari parametri:

- num_iter (numero di iterazioni per il metodo di Eulero): maggiore è il valore di questo parametro e più volte verrà applicato il metodo
- delta_t (costante d'integrazione): maggiore è il valore di questo parametro e maggiore sarà l'intensità con cui viene applicato il metodo
- c (coefficiente di controllo del gradiente): permette di far risaltare di più (se $c < 1$) o di meno (se $c > 1$) i bordi
- sigma (costante di controllo diffusione uniforme): il calcolo del vettore c utilizzato per modificare l'equazione del calore avviene tramite gradiente. Come già analizzato nella sezione 5.3, conviene fare questo calcolo su una versione sfocata dell'immagine, per fare ciò applichiamo l'equazione del calore. Modificarne questo coefficiente significa modificare l'intensità con cui viene applicata. Un valore più alto sfoca di più e porta ad eliminare più rapidamente il rumore a discapito della definizione dei bordi.

Definiremo dei parametri che andranno a costituire il nostro standard ed apporteremo delle modifiche per vedere chiaramente la differenza tra i risultati ottenuti. Vediamo quindi alcuni esempi.

3.1.1 Esempio 1 - Risultati ottenuti con i parametri standard

Vediamo innanzitutto degli esempi di casi reali, Vediamo come operano i due filtri implementati con i parametri mostrati nel codice alla fine dello scorso capitolo Parametri:

- num_iter=20
- delta_t=0.1
- c=60
- sigma=1



Figura 3.1. Da sinistra a destra: immagine con rumore, immagine filtrata con equazione del calore e immagine filtrata con metodo Perona-Malik.

Per brevità d'ora in avanti ci riferiremo a questi parametri come parametri standard.

3.1.2 Esempio 2 - Variazione del numero di iterazioni

Andiamo adesso a vedere come il numero di iterazioni influisce sul risultato filtriamo la stessa immagine confrontando i risultati dopo 10 iterazioni, dopo 20 iterazioni (valore standard) e dopo 100 iterazioni. Parametri:

- num_iter=10, 20, 100
- delta_t=0.1
- c=60
- sigma=1



Figura 3.2. In ordine: immagine con rumore e immagine filtrata dopo 10 iterazioni, dopo 20 iterazioni e dopo 100 iterazioni.

Per poter meglio apprezzare le differenze vediamo delle sezioni d'immagine

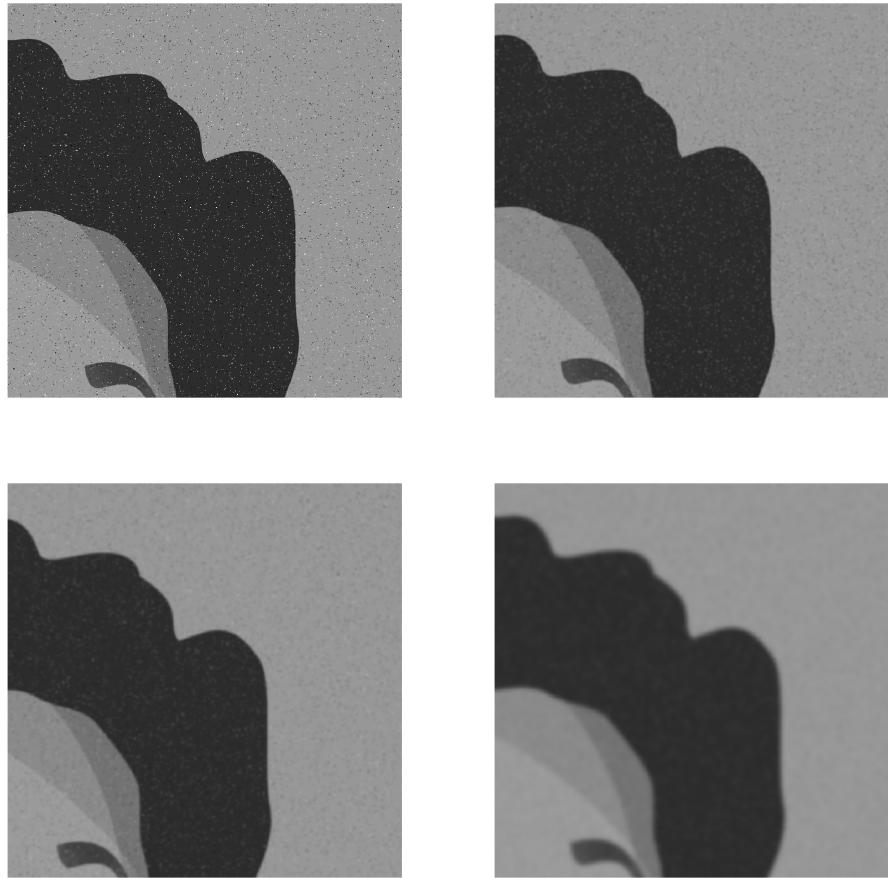


Figura 3.3. In ordine: immagine con rumore e immagine filtrata dopo 10 iterazioni, dopo 20 iterazioni e dopo 100 iterazioni.

Tramite queste immagini possiamo apprezzare come ad ogni ulteriore iterazione il rumore venga ridotto, rovinando tuttavia i bordi. È dunque bene valutare quando sia il caso di aumentare il numero di iterazioni e quando no. Se l'immagine in esame presenta dei bordi molto marcati, come ad esempio una scritta converrà preservarli il più possibile. Al contrario se l'immagine è caratterizzata per lo più da sfocature ci si può concedere di aumentare il numero di iterazioni.

È bene ricordare però che aumentare il numero di iterazioni porta un notevolmente aumento del costo computazionale e dunque del tempo di elaborazione.

3.1.3 Esempio 3 - Variazione della costante d'integrazione

Passiamo ora alla costante d'integrazione e come cambiarla influisca con i risultati.
Parametri:

- num_iter=20
- delta_t=0.01, 0.1, 0.25
- c=60
- sigma=1



Figura 3.4. In ordine: immagine con rumore, immagine filtrata con $\text{delta_t}=0.01$, con $\text{delta_t}=0.1$ e con $\text{delta_t}=0.25$.

Da queste immagini è chiaramente visibile che per delta_t prossimo allo zero (in questo caso 0.01) il problema del rumore non viene risolto, Ma solo attenuato. D'altro canto aumentandone

il valore si perdono informazioni sui bordi molto più velocemente. Per poter meglio apprezzare questa differenza guardiamo delle sezioni delle ultime due immagini

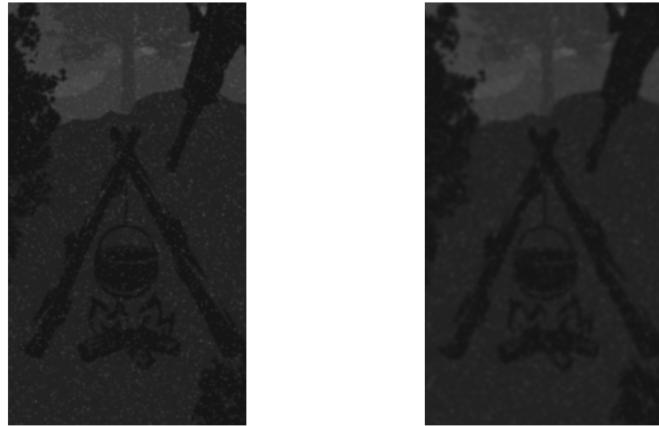


Figura 3.5. In ordine: immagine filtrata con $\text{delta_t}=0.1$, e con $\text{delta_t}=0.25$.

Vediamo quindi che aumentando questo parametro abbiamo un'immagine più sfuocata. Bisogna però fare attenzione poiché un valore anche solo di poco più alto può portare ad effetti distruttivi sull'immagine. Poniamo ad esempio $\text{delta_t}=0.3$.

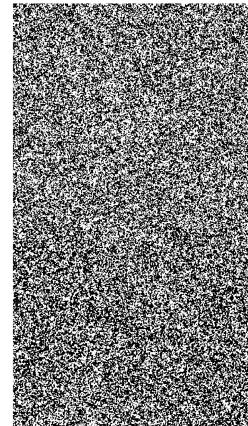


Figura 3.6. Immagine filtrata con $\text{delta_t}=0.3$.

Questa evidenza sperimentale conferma quanto visto nell'**Osservazione 2.2.1**. Per garantire la stabilità numerica occorre quindi utilizzare valori $\text{delta_t} \in [0, \frac{1}{4}]$.

3.1.4 Esempio 4 - Variazione della costante di controllo

Adesso vediamo la costante c di controllo, la quale influenza il coefficiente c che è ciò che caratterizza il metodo differenziandolo dalla sola applicazione dell'equazione del calore. Vediamo come questo parametro regola in che misura i bordi verranno preservati e come cambiarla influisca con i risultati.

Parametri:

- num_iter=20
- delta_t=0.1
- $c=6, 20, 60$
- sigma=1



Figura 3.7. In ordine: immagine con rumore, immagine filtrata con $c=6$, con $c=20$ e con $c=60$.

Questi esempi ci permettono di apprezzare come al crescere di c abbiamo una maggior riduzione del rumore, mentre i bordi vengono preservati meno.

Analiticamente $c = c(|\frac{\nabla(u)}{k}|^2) \Rightarrow$ al crescere i bordi vengono preservati meno.

Casi limite: c alto

- num_iter=20
- delta_t=0.1
- c=60000
- sigma=1



Figura 3.8. In ordine: immagine filtrata con $c=60000$ e immagine filtrata con equazione del calore.

Analogamente, con un valore molto alto possiamo vedere che l'immagine sembra solamente diffusa, e infatti $c = c(|\frac{\nabla(u)}{k}|^2) \Rightarrow \lim_{k \rightarrow \infty} c(|\frac{\nabla(u)}{k}|^2) = 1$.

Ma allora $\frac{\partial u}{\partial t}(t, x) = \operatorname{div}(c \nabla(u)) = \operatorname{div}(\nabla(u)) = \Delta(u)$ ritrovando quindi l'equazione del calore, numericamente infatti troviamo: $\min(\min([cN, cS, cW, cE])) = 0.999996871393924$ e $\max(\max([cN, cS, cW, cE])) = 1$ riconosciamo quindi un appiattimento di c ad 1.

Casi limite: c basso

Parametri:

- num_iter=20
- delta_t=0.1
- c=0.001
- sigma=1

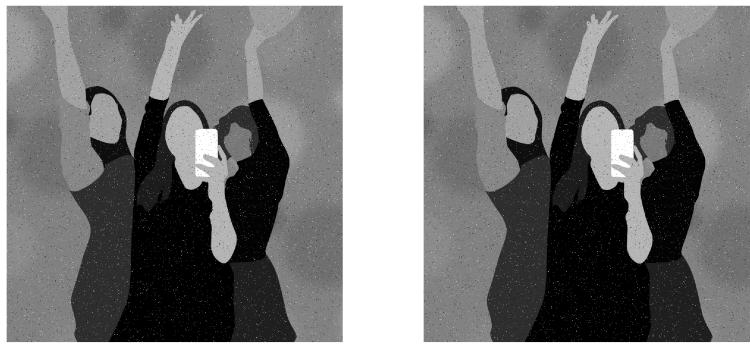


Figura 3.9. In ordine: immagine con rumore e immagine filtrata con $c=0.001$.

Possiamo facilmente notare come con un valore molto basso di c l'immagine sembra non subire variazioni, effettivamente $c = c(|\frac{\nabla(u)}{k}|^2) \Rightarrow \lim_{k \rightarrow 0} c(|\frac{\nabla(u)}{k}|^2) = 0$.

Ma allora $\frac{\partial u}{\partial t}(t, x) = \operatorname{div}(c \nabla(u)) = \operatorname{div}(0 * \nabla(u)) = 0$, ma $\frac{\partial u}{\partial t}(t, x) = 0 \Leftrightarrow u = \text{cost}$ cioè non c'è variazione.

In questo caso non riportiamo i dati sul massimo e sul minimo di c perché anche se ci aspettiamo un appiattimento sullo 0 ci sarà sicuramente qualche punto in cui non c'è variazione di colore la derivata sarà quindi nulla e per quanto piccolo possiamo scegliere k il rapporto farà comunque 0 e quindi c sarà 1. Questo non ci preoccupa siccome il metodo sfocherà una porzione d'immagine di colore costante per cui anche in questi punti non avremo nessuna variazione

3.1.5 Esempio 5 - Variazione della costante di diffusione

Adesso vediamo la costante Sigma, la quale regola l'applicazione dell'equazione del calore che influenza il coefficiente c . Vediamo quindi anche in questo caso come questo parametro regola in che misura i bordi verranno preservati e come cambiarla influisca con i risultati.

Parametri:

- num_iter=20
- delta_t=0.1
- c=60
- sigma=0.5, 1, 2

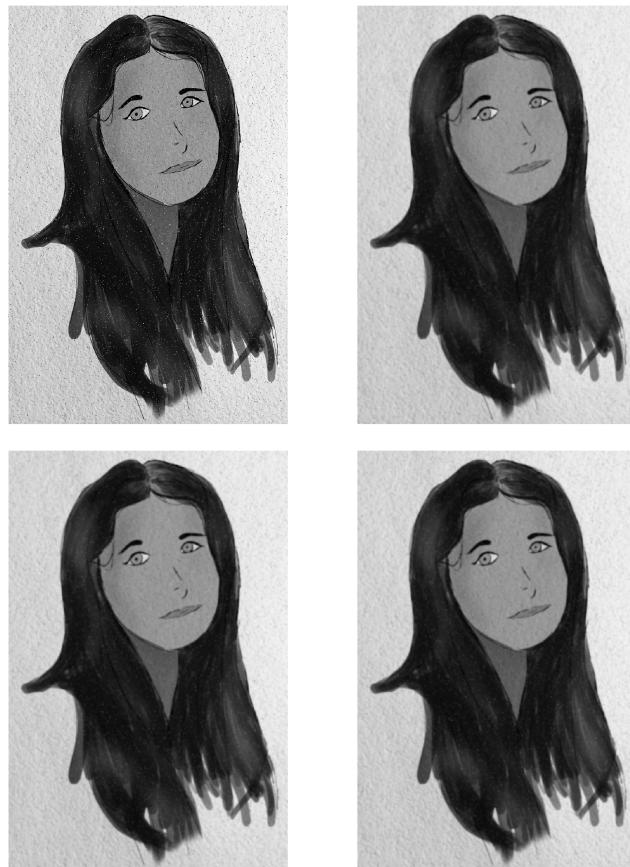


Figura 3.10. In ordine: immagine con rumore, immagine filtrata con $\sigma=0.5$, con $\sigma=1$ e con $\sigma=2$.

Questi esempi però non ci permettono di apprezzare una gran differenza, infatti sigma influenza il gradiente, questo va però diviso per c che in questo caso è 60 appiattendo i valori del gradiente.

Infatti, $\frac{\nabla(u)_1}{k} - \frac{\nabla(u)_2}{k} = \frac{\nabla(u)_1 - \nabla(u)_2}{k}$ cioè anche la differenza tra i due gradienti sarà 60 volte minore e quindi poco visibile.

Per rendere più evidenti queste differenze possiamo allora abbassare il valore di k.

Parametri:

- num_iter=20
- delta_t=0.1
- c=0.5
- sigma=0.5, 1, 2

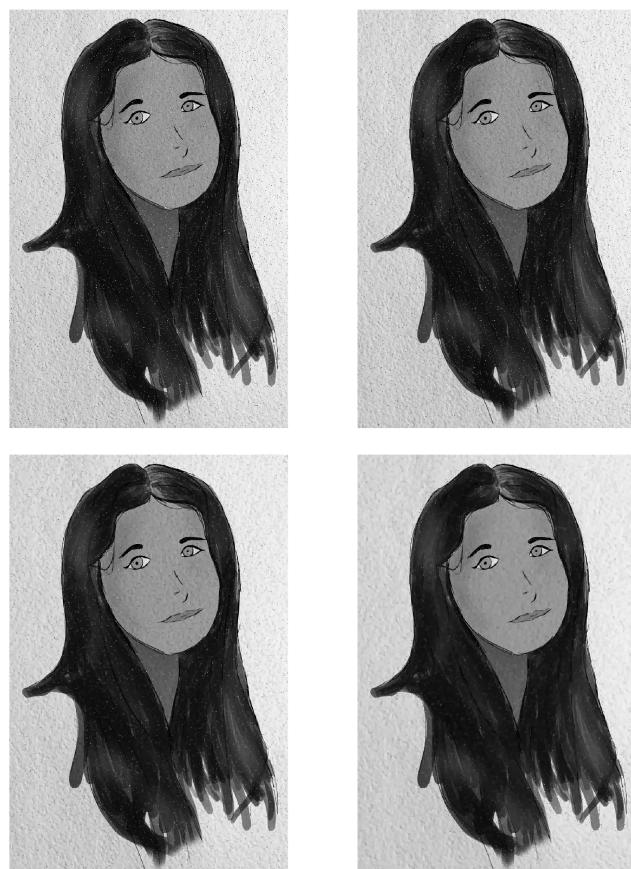


Figura 3.11. In ordine: immagine con rumore, immagine filtrata con sigma=0.5, con sigma=1 e con sigma=2.

Anche in questo caso, come visto nell'esempio 3, bisogna fare attenzione a non impostare valori troppo alti. Nell'esempio 3 ce ne siamo preoccupati perché il metodo prevede il calcolo $\text{diff_im} = \text{diff_im} + \text{delta_t} * (\text{cN}.*\text{nablaN} + \text{cS}.*\text{nablaS} + \text{cW}.*\text{nablaW} + \text{cE}.*\text{nablaE})$ e si era detto che per garantire la stabilità numerica utilizziamo $\text{delta_t} \in [0, \frac{1}{4}]$. Ricordiamo adesso che per operare il filtraggio richiamiamo l'equazione del calore, che invece prevede il calcolo $u = u + \text{sigma} * \text{dt} * (u_{xx} + u_{yy})$, per cui per assicurare la stabilità numerica imponiamo anche $\text{delta_t} * \text{sigma} \in [0, \frac{1}{4}]$, in accordo con quanto visto nella

Proposizione 2.1.1. Vediamo ad esempio quali risultati otteniamo ponendo $\text{sigma}=2$, che abbiamo visto dare un buon risultato, e $\text{delta_t}=0.2$.

Parametri:

- num_iter=20
- delta_t=0.2
- c=0.5
- sigma=2

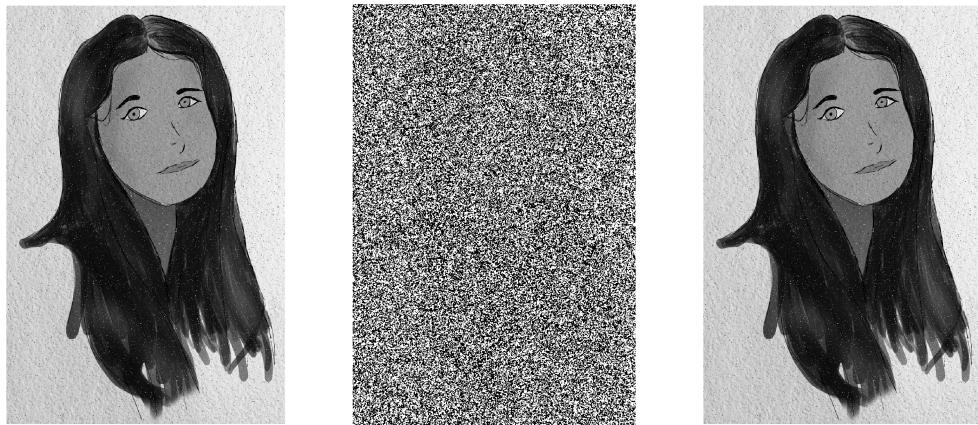


Figura 3.12. In ordine: immagine originale, immagine filtrata con equazione del calore e immagine filtrata con $\text{sigma}=2$.

Otteniamo quindi un'immagine quasi non filtrata, numericamente infatti troviamo: $\min(\min([\text{cN}, \text{cS}, \text{cW}, \text{cE}]))=0$ e $\max(\max([\text{cN}, \text{cS}, \text{cW}, \text{cE}]))=4.3775e-106$ riconosciamo quindi un appiattimento di c sullo 0.

3.2 Conclusioni

La trattazione svolta ha permesso di utilizzare alcuni risultati di diverse branche della matematica per approcciarsi ad un problema reale. Si sono visti i concetti basilari che permettono l’elaborazione digitale delle immagini tramite un calcolatore quali i concetti di risoluzione, pixel e spazio di memoria occupato dalle immagini al variare di risoluzione e numero dei colori. Si sono poi introdotti i concetti di convoluzione e soluzione fondamentale di un operatore differenziale, strumenti imprescindibili nello studio delle equazioni alle derivate parziali. Sono stati introdotti alcuni semplici problemi di filtraggio digitale risolubili tramite strumenti di algebra lineare. Sono stati altresì approfonditi i metodi numerici, come il metodo delle differenze finite con relativo errore di troncamento e il metodo di Eulero esplicito, necessari all’implementazione del metodo Perona-Malik, ricavandone anche le condizioni di stabilità. Lungo la trattazione sono stati inoltre richiamati i concetti di analisi matematica utilizzati, come derivate, gradiente e laplaciano. Grazie a questi concetti matematici è stato possibile, sfruttando anche strumenti tipici dell’elaborazione digitale delle immagini come le maschere di convoluzione, ottenere risultati semplici come la diffusione di un’immagine tramite equazione del calore e rilevamento dei bordi, che hanno fatto da base per l’implementazione del filtro di diffusione anisotropica in esame. In fine sono stati mostrati diversi esempi applicativi che mostrano l’operato dello script MATLAB ottenuto.

Lo studio di questo metodo ha storicamente aperto le porte a successivi studi in questo campo, portando allo sviluppo di numerosi modelli di diffusione non lineare. Di particolare rilievo è il metodo proposto da Cottet e Germain nel 1993¹, che si differenzia dal metodo Perona-Malik adottando una diffusione regolata da un sistema di tensori portando ad una diffusione diversa lungo le diverse direzioni definite dalla geometria locale dell’immagine, ottenendo quindi un metodo di diffusione anisotropica in senso proprio. In questo approccio, l’introduzione di una scala di integrazione nel tensore di struttura è una caratteristica essenziale del modello. Successivamente (nel 1996) Cottet e El Ayyadi hanno proposto un modello di restauro dell’immagine modificato che sostituisce la regolarizzazione spaziale con una regolazione temporale.

Un altro tipo di schema di filtraggio anisotropico è stato proposto da Yang² e i suoi collaboratori. Invece di utilizzare i gradienti locali come mezzo per controllare l’anisotropia del filtro, con il metodo di Yang si costruisce il kernel del filtro in base alle caratteristiche anisotropiche locali dell’immagine e il loro schema di stima dell’orientamento si basa sul fatto che lo spettro di potenza di un pattern orientato giace lungo una linea che passa per l’origine di un dominio di Fourier. Tuttavia le stime dei parametri nello schema di filtraggio di Yang sono non sono ottimali per le caratteristiche delle immagini, come angoli, giunzioni o bordi.

¹Weickert [1999]

²Yang et al. [1996]

Bibliografia

Rorres C. Anton H. *Elementary linear algebra: Applications version.* 2010.

Marco Fodde. La fotografia in bianconero. *Roma, Edizioni Reflex*, 2000.

Rafael C. Gonzalez and Richard E. Wood. *Digital Image Processing.* Addison-Wesley, 1992.

Giovanni Monegato. *Fondamenti di Calcolo Numerico.* 1998.

Saleri Quarteroni, Sacco. *Numerical Mathematics Springer Texts in Applied Mathematics.* 2010.

Gerald W. Recktenwald. Finite-difference approximations to the heat equation. 2004.

Joachim Weickert. Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 1999.

G.Z. Yang, P. Burger, D.N. Firmin, and S.R. Underwood. Structure adaptive anisotropic image filtering. *Image and Vision Computing*, 1996.