

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. Nome Cognome

prof. Nome Cognome

firma dei relatori

.....
.....

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 dai primi tempi le ha utilizzate per ricordare, illustrare, comunicare, etc. Per questo motivo, cercare di sfruttarle nel migliore dei modi è oggigiorno un problema di grande interesse. Molteplici sono i metodi di elaborazione digitali detti "di filtraggio", il cui scopo 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 (nel quale ci si sta introducendo), 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 l'implementazione del 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

I candidati ringraziano vivamente il Granduca di Toscana per i mezzi messi loro a disposizione, ed il signor Von Braun, assistente del prof. Albert Einstein, per le informazioni riservate che egli ha gentilmente fornito loro, e per le utili discussioni che hanno permesso ai candidati di evitare di riscoprire l'acqua calda.

Questa parte è rimasta inalterata da com'era nel template.

Indice

Elenco delle figure

*If you cannot understand my
argument, and declare
it's Greek to me
you are quoting Shakespeare.*

[B. LEVIN, Quoting Shakespeare]

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 nota come "*altissimo contrasto*" permetteva di avere foto con soli bianchi e soli neri, 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 20. 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.

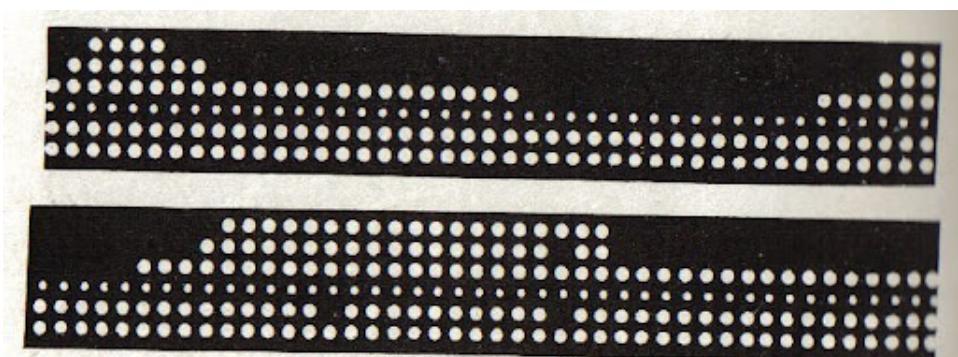


Figura 1.1. Nastro dati codificante un'immagine.

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.

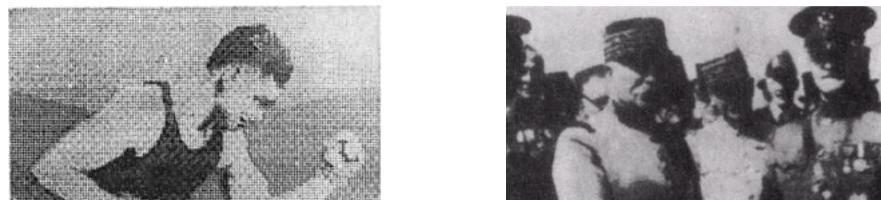


Figura 1.2. La prima immagine risale al 1921 ed è composta da 5 gradazioni di grigio. La seconda è del 1929 ed è composta da 15 gradazioni di grigio.

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

? Nel 1960 presso il Jet Propulsion Laboratory, il Massachusetts Institute of Technology, i Bell Laboratories, l’Università del Maryland, e altre strutture di ricerca svilupparono molte delle tecniche di elaborazione digitale di immagini (o della trasformazione di immagini digitali come spesso veniva chiamata) al fine di evitare le debolezze operative delle fotocamere a pellicola per missioni scientifiche e militari.

Queste trovarono poi applicazione in immagini satellitari, immagini medicali, videocitofono, riconoscimento ottico dei caratteri, e miglioramenti fotografici.

Il costo dell’elaborazione in quel periodo era piuttosto alto con l’apparecchiatura di elaborazione. Le cose cambiarono negli anni settanta, quando computer più economici e hardware dedicato furono resi disponibili sul mercato.

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

Negli anni 2000, grazie all’avvento di computer più veloci, l’elaborazione delle immagini digitali diventò la forma più comune di elaborazione delle immagini e, in generale, divenne il metodo più utilizzato data la sua versatilità e il basso costo.

La tecnologia di elaborazione delle immagini digitali per applicazioni mediche è stata inserita nel 1994 nella Hall of Fame della Space Foundation.

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 cosa sia un oggetto, per poterlo poi 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 quindi su superfici, cioè oggetti bidimensionali, di dimensioni finite e le vediamo perchè i nostri occhi percepiscono il susseguirsi di colori diversi. Come codificare tali entità? Come per tutti gli oggetti reali, sebbene abbiano dimensioni finite, le immagini sono distribuzioni continue, o per meglio dire, il susseguirsi delle gradazioni di colore avviene in una maniera che possiamo considerare come continua. Questo è il primo problema che ci si pone quando si pensa a come codificare delle immagini.

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**, assegname 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 1.3. Confronto tra immagine originale e immagine codificata utilizzando una griglia 4x4.



Figura 1.4. 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, come anticipato, 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.

Volendo definire in maniera più precisa che cosa è un'immagine digitale, diremmo che quest'ultima è una funzione da \mathbb{R}^2 in \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 da \mathbb{R}^2 in \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.

1.2.2 Cosa sono i filtri

Una volta capito che un'immagine è una funzione possiamo definire un filtro come una seconda funzione che convoluta alla prima da il risultato richiesto.

Una equazione alle derivate parziali (PDE) esprime una evoluzione. Sia u la nostra immagine e u_0 lo stato in cui si trova inizialmente, allora per convoluzione possiamo dire che

$$u(x) = \frac{1}{w(x)} \int \int d(x - \xi) \tilde{d}(u_0(x) - u_0(\xi)) u_0(\xi) d\xi. \quad (1.1)$$

$$\text{con } w(x) = \int \int d(x - \xi) \tilde{d}(u_0(x) - u_0(\xi)) d\xi ?$$

In matematica, la **convoluzione** è un'operazione tra due funzioni di una variabile che consiste nell'integrare il prodotto tra la prima e la seconda traslata di un certo valore.

E questo è a tutti gli effetti un filtro. 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 anzichè 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**. Questi sono molto utili per introdurre i primi concetti matematici, come l'impiego di matrici.

Definiamo una **matrice di trasformazione** che moltiplicata per un vettore di coordinate ci restituisca un altro vettore di coordinate. Questo vuol dire che quel punto va spostato dalle coordinate in cui si trovava a quelle appena calcolate. Ad esempio:

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

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

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

Cioè ogni punto rimane alla stessa quota ma cambia la propria x con $-x$, che è esattamente ciò che si intende per riflessione lungo l'asse verticale.

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

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 e si definiscono sull'immagine di destinazione i corrispondenti punti e di conseguenza le corrispondenti linee.



Figura 1.5. Processo di morphing con alcuni risultati intermedi.

Durante la dissolvenza dall'immagine iniziale a quella finale, le immagini vengono deformate 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.

Capitolo 2

Trattazione del problema

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. 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. 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:

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

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

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

Consideriamo adesso gli sviluppi di Taylor che hanno portato a queste due approssimazioni e sottraiamo membro a membro.

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

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, allora quest'ultima può essere approssimata nel seguente modo:

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

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 del 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.

Per definizione, l'errore è la differenza 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

- Differenza finita in avanti

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

$$\downarrow$$

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

- Differenza finita all'indietro

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

$$\downarrow$$

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

- Differenza finita centrale

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

$$\downarrow$$

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

Allo stesso modo, applicando la definizione di errore per la derivata seconda:

$$\left. \frac{d^2u}{dx^2} \right|_{x_i} - \frac{u_{i+1} - u_i}{\Delta(x)} \approx \frac{\Delta(x)^2}{2} \left. \frac{d^2u}{dx^2} \right|_{\xi}.$$

Risulta quindi evidente che l'errore dipende da $\Delta(x)^2$. 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 $\Delta(x)^3$ e non da un $\Delta(x)^2$, 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 brevità di notazione poniamo $\Delta(x) = h$.

Dagli sviluppi di Taylor in x_j di $u(x_{j\pm 1}) = u(x_j \pm h)$ fino al terzo ordine, si ottiene che

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

dove ξ_j è un punto opportuno in $(x_{j-1}, x_j + 1)$. Quindi, considerando il problema modello con condizioni di Dirichlet, 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 $\tau_j = \frac{1}{12}u(4)(\xi_j)h^2$ (errore di troncamento locale) e ponendo:

$$\mathbf{u} = [u_x \dots u_{N-1}]^T$$

$$\boldsymbol{\tau}_h = [\tau_x \dots \tau_{N-1}]^T$$

$$\mathbf{b}_h = [(f(x_1) + \frac{g_0}{h^2}, f(x_2), \dots, f(x_{N-2}), f(x_{N-1}) + \frac{g_1}{h^2}]^T$$

si può allora scrivere in forma matriciale, $A_h \mathbf{u} = \mathbf{b}_h + \boldsymbol{\tau}_h$ dove

$$A_h = \frac{1}{h^2} \text{tridiag}(-1, 2, -1) + \text{diag}(\sigma_1, \dots, \sigma_{N-1}) e \sigma_j = \sigma(x_j).$$

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.

Risultando che $\boldsymbol{\tau}_h$ tende a zero quando h tende a zero, il metodo dicesi consistente. In particolare nel caso in analisi, come osservato in partenza, risulta $\boldsymbol{\tau}_h = O(h^2)$.

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

Per studiarne la convergenza è necessario considerare il comportamento dell'errore $\mathbf{e}_h = \mathbf{u}_h - \mathbf{u}$ quando h tende a zero. Dato che risulta $A_h \mathbf{e}_h = \boldsymbol{\tau}_h$, e quindi $\mathbf{e}_h = A_h^{-1} \boldsymbol{\tau}_h$ si può quindi scrivere: $\|\mathbf{e}_h\| \geq \|A_h^{-1}\| \|\boldsymbol{\tau}_h\|$

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

A questo scopo osservando 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:

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

Per le ipotesi su σ questo implica $A_h^{-1} \leq A_{0h}^{-1}$ osserviamo che $\|A_{0h}^{-1}\| = \max_j (A_{0h}^{-1} \mathbf{e})_j$ dove \mathbf{e} 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 $A_{0h}^{-1}(\mathbf{e})_j = \phi(x_j)$ e quindi che

$$A_h^{-1} \leq A_{0h}^{-1} \leq \max_{0 < x < 1} |\phi_x|.$$

Questo risultato di stabilità ci permette di concludere che l'errore \mathbf{e}_h ha lo stesso ordine dell'errore di troncamento $\boldsymbol{\tau}_h$ e di conseguenza che il metodo è convergente del secondo ordine. Si noti che l'uniforme limitatezza della norma di A_h^{-1} implica che il metodo numerico sia stabile.

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.

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \Delta u(t, x) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}. \quad (2.1)$$

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

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

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:nx nx) - 2*u + u(:,:,1 1:nx-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. Esistono procedimenti come il metodo Perona-Malik che sono decisamente più efficaci.

2.1.3 Studio della stabilità del metodo

Procediamo adesso ad uno studio della stabilità di tale metodo tramite il **metodo di Von Neumann?**

Il metodo di Von Neuman si basa sulla decomposizione degli errori in serie di Fourier. Senza ledere 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 $r = \sigma \frac{\Delta t}{(\Delta x)^2}$

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

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

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

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

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

$$\epsilon(x, t) = \int_{k=-\frac{\pi}{\Delta x}}^{k=\frac{\pi}{\Delta x}} E_k(t) e^{ikx} dk$$

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

$$\epsilon_j^n = E_m(t) e^{ik_m x}$$
 da cui:

$$\epsilon_j^{n+1} = E_m(t + \Delta t) e^{ik_m x}$$

$$\epsilon_{j+1}^n = E_m(t) e^{ik_m (x + \Delta x)}$$

$$\epsilon_{j-1}^n = E_m(t) e^{ik_m (x - \Delta x)}$$

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:

$$E_m(t + \Delta t)e^{ik_m x} = E_m(t)e^{ik_m x} + r(E_m(t)e^{ik_m(x+\Delta x)} - 2E_m(t)e^{ik_m x} + E_m(t)e^{ik_m(x-\Delta x)})$$

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

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

$$\frac{E_m(t + \Delta t)}{E_m(t)} = 1 + r(e^{ik_m \Delta x} + e^{-ik_m \Delta x} - 2).$$

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

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

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:

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

Come detto, il metodo è stabile $\Leftrightarrow \left| \frac{E_m(t + \Delta t)}{E_m(t)} \right| \leq 1$ ma visto che $\frac{E_m(t + \Delta t)}{E_m(t)} = 1 - 4r \sin^2\left(\frac{\theta}{2}\right) \Rightarrow$ 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 $r = \sigma \frac{\Delta t}{(\Delta x)^2} \Rightarrow$ condizione sufficiente affinché il metodo sia stabile è $\sigma \frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$

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

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

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

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

Nel caso in esame $\Delta x = \Delta y = 1 \Rightarrow r_x + r_y = \sigma \Delta t + \sigma \Delta t \leq \frac{1}{2} \Rightarrow \sigma \Delta t \leq \frac{1}{4}$

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:

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \Delta u(t, x) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}. \quad (2.2)$$

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 introdurrà un coefficiente $c \operatorname{div}(c \nabla(u))$, nel caso esso sia uno scalare costante, avremo $c \operatorname{div}(c \nabla(u)) = c \operatorname{div}(\nabla(u)) = c \Delta(u)$ ed è il caso implementato nelle pagine precedenti. Il metodo Perona-Malik invece prevede l'impegno di un coefficiente c non costante, ma che cambi a seconda del pixel su cui si opera. In questo senso, questo filtro viene detto di diffusione anisotropa. Il termine anisotropo indica come la diffusione sia diversa nelle varie direzioni, capiamo quindi come questa notazione, per quanto esplicativa sia in verità un abuso di notazione siccome la diffusione non avviene in direzioni diverse ma semplicemente con intensità diverse. Esistono tuttavia dei metodi metodi di diffusione anisotropa, in tal caso c sarà un tensore variabile.

Ritornando al metodo in esame, il problema affrontato diventa

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) - \operatorname{div}(c \nabla(u)) = 0 & x \in \mathbb{R}^2, t \geq 0 \\ \frac{\partial u}{\partial N} = 0 & x \in \mathbb{R}^2, t \geq 0 \\ u(0, x) = u_0(x) \end{cases}. \quad (2.3)$$

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

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

$$\frac{\partial u}{\partial t}(t, x) = \operatorname{div}(c \nabla(u)) = \quad (2.4)$$

$$= \frac{c \nabla u}{\partial x} + \frac{c \nabla u}{\partial y} \quad (2.5)$$

$$\approx \frac{k_N \nabla_N + k_S \nabla_S}{2} + \frac{k_W \nabla_W + k_E \nabla_E}{2} \quad (2.6)$$

$$= \frac{1}{2}(k_N \nabla_N + k_S \nabla_S + k_W \nabla_W + k_E \nabla_E) \quad (2.7)$$

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

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

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

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

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

Stabilità del metodo

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 $\Leftrightarrow 4 \frac{1}{2} \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$.

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

$$\Delta t \leq \frac{1}{4}.$$

2.2.2 Maschere di convoluzione

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

Un filtro è una funzione $F : \mathbb{R}^2 \rightarrow \mathbb{R}$, in quanto tale avrà un nucleo $\text{Ker}(F) := \{v \in \mathbb{R}^2 : Av = 0\}$ che sarà un polinomio a due incognite. Date in input a tale polinomio delle coordinate, questo 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

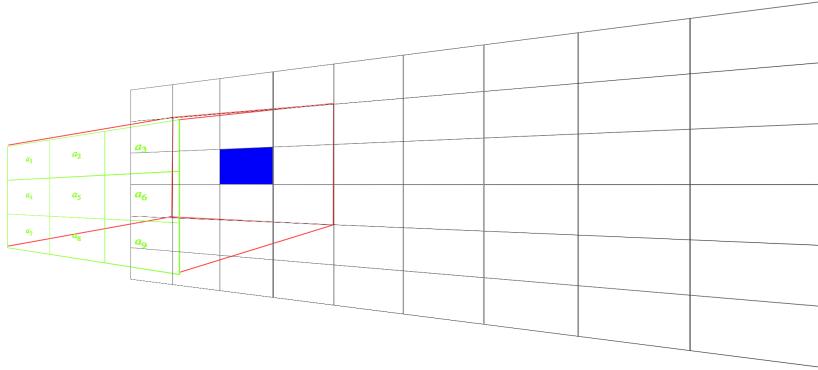


Figura 2.3. Griglia dell'immagine in nero, maschera in verde, pixel in esame in blu

Per ogni pixel ne viene ricalcolato il valore come segue

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

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

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 $u(p_{i,j}) * a = p_{i,j+1} - p_{i,j}$ 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

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.

Si riporta un piccolo script MATLAB che mette in evidenza questo aspetto.

```

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)')

```

Lo script permette di osservare come con diverse immagini molto semplici se otteniamo i risultati attesi.

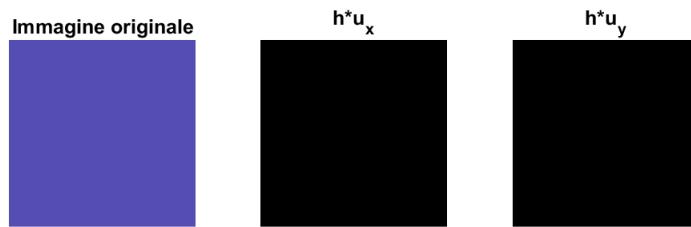
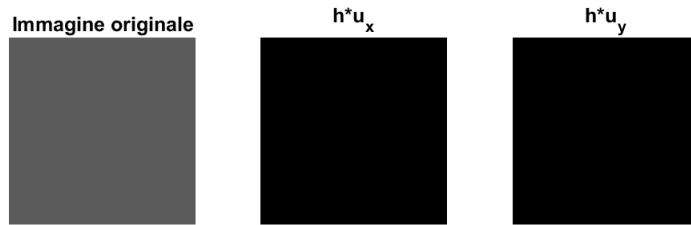


Figura 2.5. Derivate parziali di una tinta unita.

Si può vedere come con un’immagine a tinta unita (che sia essa in bianco e nero o a colori) le derivate sono nulle, quindi lo saranno anche gradiente e laplaciano. Confermiamo inoltre che questi principi valgono sia a colori che in bianco e nero.

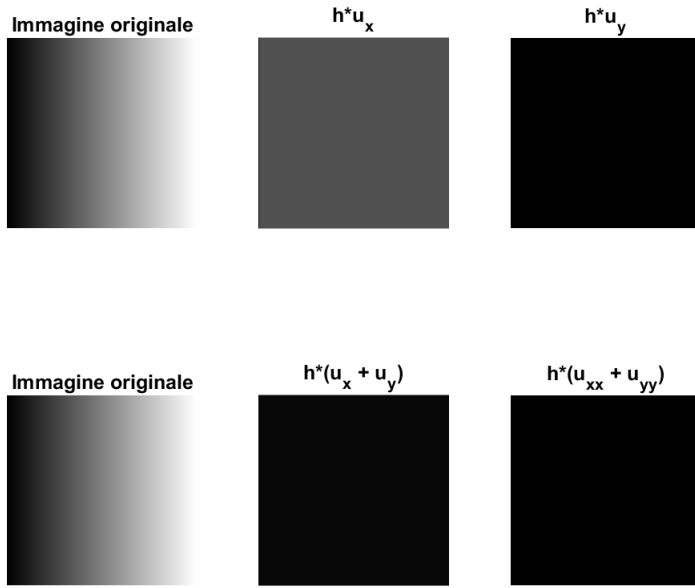


Figura 2.6. Derivate parziali, gradiente e laplaciano di una sfumatura orizzontale.

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 perché 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.

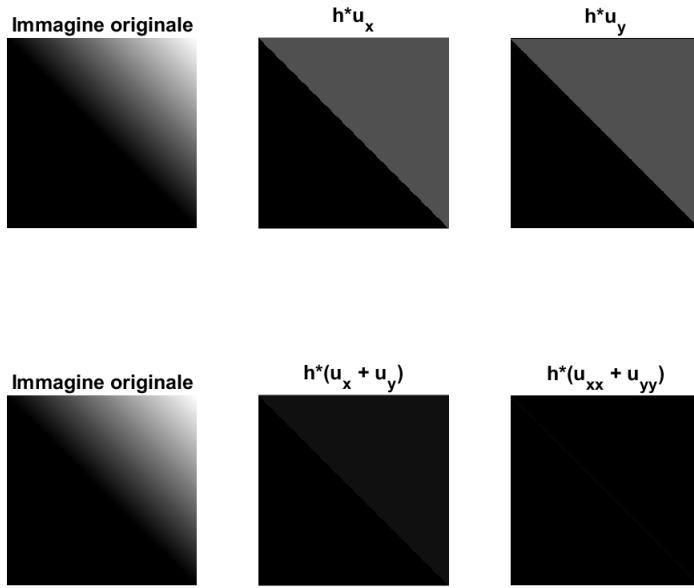


Figura 2.7. Derivate parziali, gradiente e laplaciano di una sfumatura diagonale.

Presi 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 riporta un ultimo esempio, provando ad introdurre una semplice figura e osserviamo cosa accade.

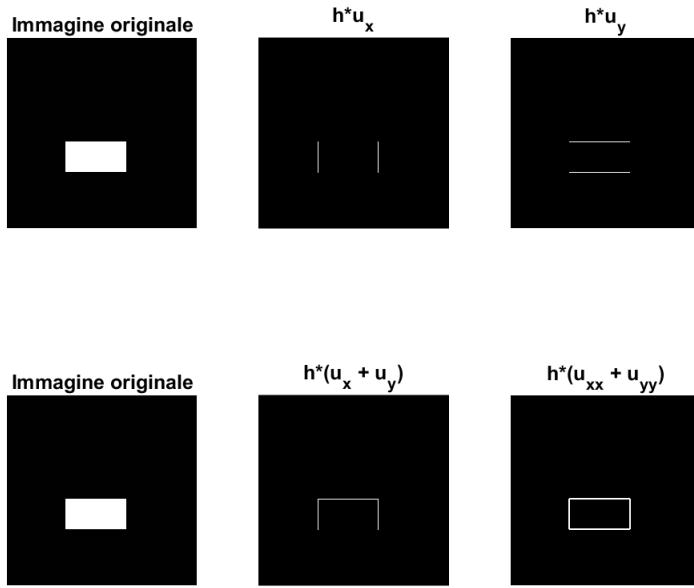


Figura 2.8. Derivate parziali, gradiente e laplaciano di una figura semplice.

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 perchè. 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.

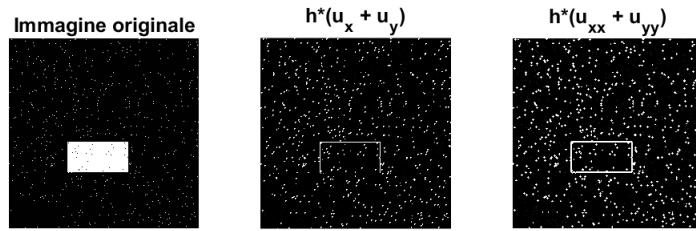
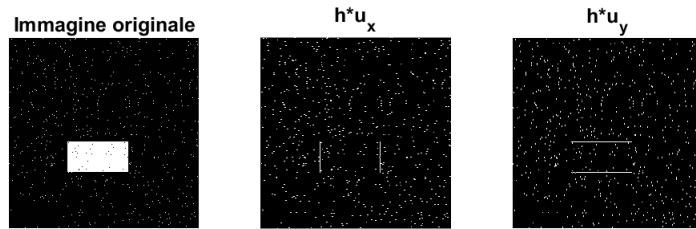


Figura 2.9. Derivate parziali, gradiente e laplaciano di una figura semplice con rumore.

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 rovati, 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 $c(0)=1$ (quindi dove non c'è bordo opera l'eq del calore) e $\lim_{s \rightarrow \infty} c(s) = 0$ (quindi tanto più i bordi sono accentuati, tanto più l'eq del calore non deve operare). Facciamo questo banalmente perché considerando semplicemente i bordi, avremmo l'effetto opposto! Infatti dove non ci sono bordi $s = 0$ e quindi non ci sarebbe diffusione, dove ci sono bordi molto marcati $s \rightarrow \infty$ 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 $c = \frac{1}{1+s/k}$ dove k è un fattore di controllo. Scelte decisamente più classiche ed impiegate sono:

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

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.

```
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 kappa=60; %coefficiente di controllo del
12 %gradiente
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 $\kappa < 1$) o di meno (se $\kappa > 1$) 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. 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.

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     cN = exp(-(nablaN_blur/kappa).^2);
39     cS = exp(-(nablaS_blur/kappa).^2);
40     cW = exp(-(nablaW_blur/kappa).^2);
41     cE = exp(-(nablaE_blur/kappa).^2);
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
746
746
747
748
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
766
766
767
768
768
769
769
770
771
772
773
774
775
776
776
777
778
778
779
779
780
781
782
783
784
785
785
786
786
787
787
788
788
789
789
790
791
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
156
```

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
- kappa (coefficiente di controllo del gradiente): permette di far risaltare di più (se $\kappa < 1$) o di meno (se $\kappa > 1$) 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

Vediamo inanzitutto un paio di esempi di casi reali ottenuti con i parametri mostrati nel codice alla fine dello scorso capitolo

Parametri:

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



Figura 3.1. Da sinistra a destra: immagine con rumore e immagine filtrata.

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

3.1.2 Esempio 2

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
- kappa=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

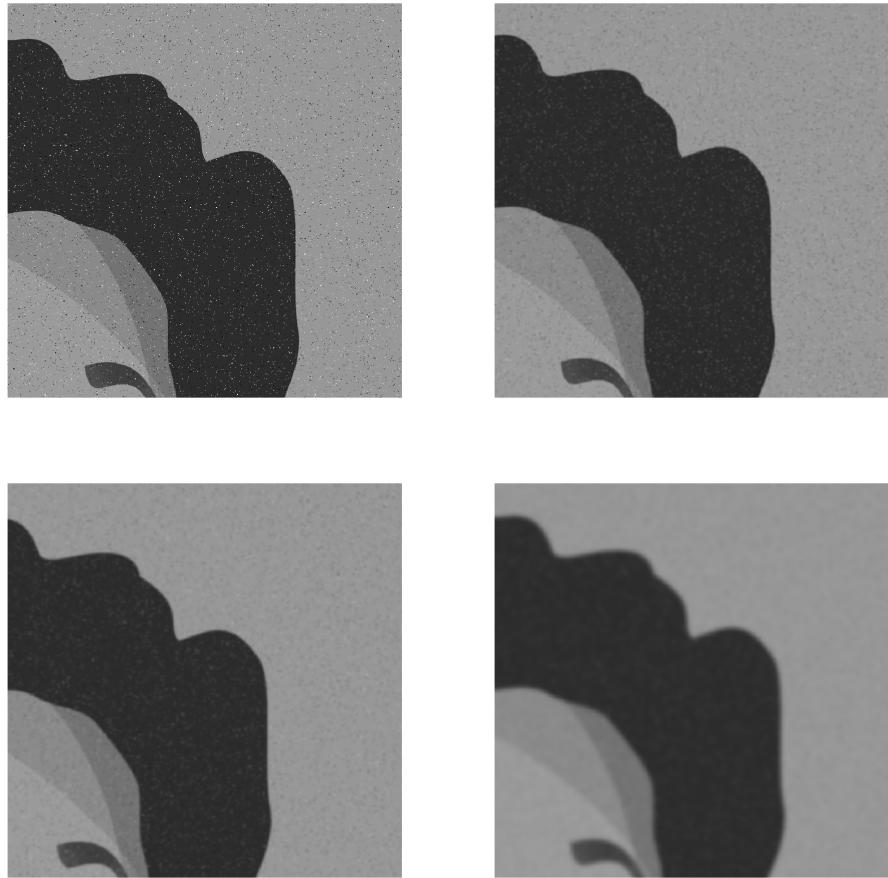


Figura 3.3. In ordine: immagine con rumore e immagine filtrata dopo 10 iterazioni, dopo 20 iterazioni e dopo 100 iterazioni.

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

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
- kappa=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, la 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

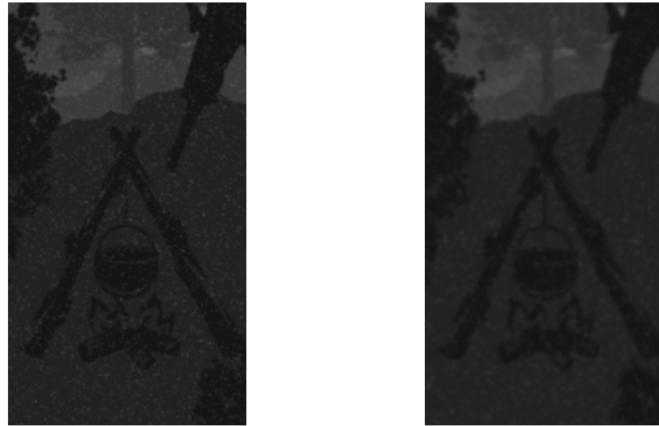


Figura 3.5. In ordine: immagine filtrata con $\text{delta_t}=0.1$, e con $\text{delta_t}=0.25$.

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 $\text{delta_t}=0.3$.

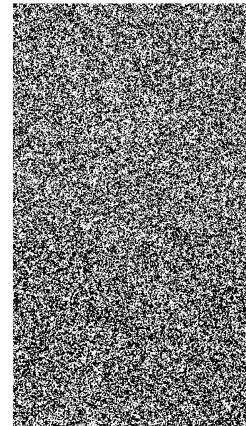


Figura 3.6. Immagine filtrata con $\text{delta_t}=0.3$.

Per garantire la stabilità numerica utilizziamo quindi valori $\text{delta_t} \in [0, \frac{1}{4}]$.

3.1.4 Esempio 4

Adesso vediamo la costante kappa 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
- kappa=6, 20, 60
- sigma=1



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

Questi esempi ci permettono di apprezzare come al crescere di κ 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: kappa alto

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



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

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

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: kappa basso

Parametri:

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



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

Possiamo facilmente notare come con un valore molto basso di κ 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

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
- kappa=60
- sigma=0.5, 1, 2

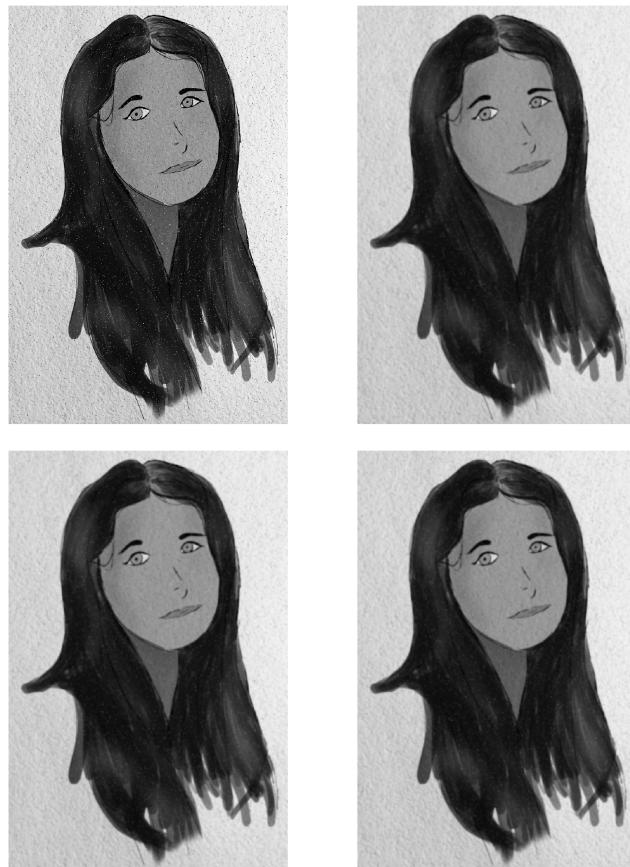


Figura 3.10. In ordine: immagine con rumore, immagine filtrata con $\sigma=0.5$, con $\sigma=1$ e con $\sigma=2$.

Questi esempi però non ci permettono di apprezzare una gran differenza, infatti sigma influenza il gradiente, questo va però diviso per kappa 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
- kappa=0.5
- sigma=0.5, 1, 2

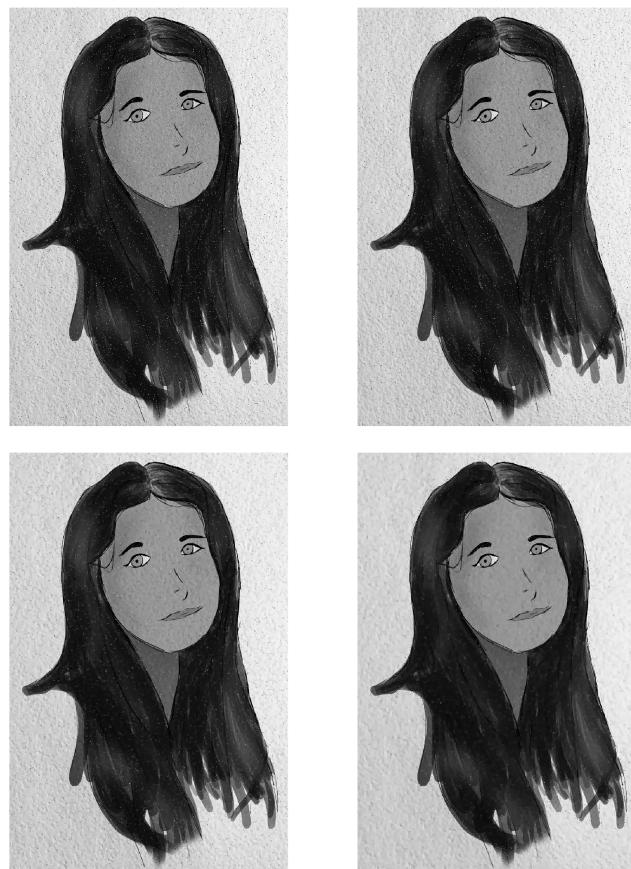


Figura 3.11. In ordine: immagine con rumore, immagine filtrata con sigma=0.5, con sigma=1 e con sigma=2.

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 utilizziamo $\text{delta_t} * \text{sigma} \in [0, \frac{1}{4}]$. 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
- kappa=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 $\max(\max([\text{cN}, \text{cS}, \text{cW}, \text{cE}]))=4.3775\text{e-}106$ riconosciamo quindi un appiattimento di c sullo 0.