Implementazione del metodo Canny Edge-detector per il rilevamento dei tratti significativi dei volti e loro ricostruzione attraverso la tecnica dell'interpolazione.

Progetto Fondamenti di Calcolo Numerico A.A. 2020/2021

Fratta Gennaro, Lenzi Francesco, Rigamonti Riccardo

Maggio 2021

Indice

1 Introduzione

La segmentazione di un'immagine nell'elaborazione digitale delle immagini è il processo di partizione di un'immagine in regioni significative. La segmentazione è di solito utilizzata per localizzare oggetti e bordi; più precisamente, è il processo con il quale si classificano i pixel dell'immagine che hanno caratteristiche comuni, pertanto ciascun pixel in una regione è simile ad altri presenti nella stessa regione relativamente ad una determinata proprietà o caratteristica (colore, intensità o texture). Regioni adiacenti sono significativamente differenti rispetto ad almeno una di queste caratteristiche. Il risultato di un'immagine segmentata è un insieme di segmenti che, collettivamente, coprono l'intera immagine. Il livello cui deve essere condotta la suddivisione dipende dal problema da risolvere, vale a dire il processo termina non appena si riescono ad isolare gli oggetti di interesse. Ottenere una buona segmentazione dell'immagine è uno dei compiti più difficili dell'Image Processing e la sua accuratezza influenza considerevolmente l'eventuale successo o fallimento di procedure di analisi computerizzata che ne fanno uso. La segmentazione può fornire in uscita la mappa dei contorni, i cui elementi non nulli identificano i contorni tra gli oggetti. Data l'eterogeneità delle possibili applicazioni, non esiste una tecnica di segmentazione che sia universalmente buona. Esistono però concetti di base del trattamento di immagini che sono comuni a molti approcci. Di fatto, gli algoritmi di segmentazione si basano su due proprietà fondamentali delle immagini: discontinuità e similarità. Nella prima categoria ricadono quelle tecniche che realizzano la partizione basandosi su repentini cambiamenti della luminosità, mentre nel secondo caso l'approccio è quello di suddividere l'immagine in regioni che sono simili sulla base di un fissato criterio. Questa trattazione si limiterà alle sole tecniche Edge-Based, ed in particolare verrà analizzato l'algoritmo di Canny per il rilevamento dei bordi dei tratti significativi dei volti. Le tecniche Edge-Based si basano sull'idea che nel passare da una regione d'interesse ad un'altra si riscontrano rapidi cambiamenti nei valori di intensità. Lo scopo dunque è il rilevamento (e la classificazione) delle discontinuità (Edges) dell'immagine (livelli di grigio).

2 Immagine

Un'immagine è una funzione che rappresenta la misurazione di alcune caratteristiche (intensità luminosa, colore, ...) di una scena; è intrinsecamente bidimensionale mentre la scena risulta 3D. Un'immagine è di norma rappresentata come una matrice di elementi discreti, detti pixel, ciascuno di un colore. Viene dunque definita come la funzione x(m,n) dell'intensità luminosa il cui valore o ampiezza, corrispondente ad una generica coppia di coordinate spaziali (m,n), indica il valore dell'intensità luminosa nel punto corrispondente. La funzione x(m,n) non è necessariamente scalare. Si consideri ad esempio il caso delle immagini a colori. E' noto infatti, che ogni colore può essere rappresentato come combinazione di una serie di colori fondamentali. Lo spazio RGB viene rappresentato come un cubo, in cui lo spigolo in basso a sinistra rappresenta il nero, quello opposto il bianco, i grigi si trovano sul segmento che unisce il nero ed il bianco ed un generico colore sarà rappresentato da un punto all'interno del cubo. L'occhio umano percepisce la somma delle 3 componenti come un unico colore.

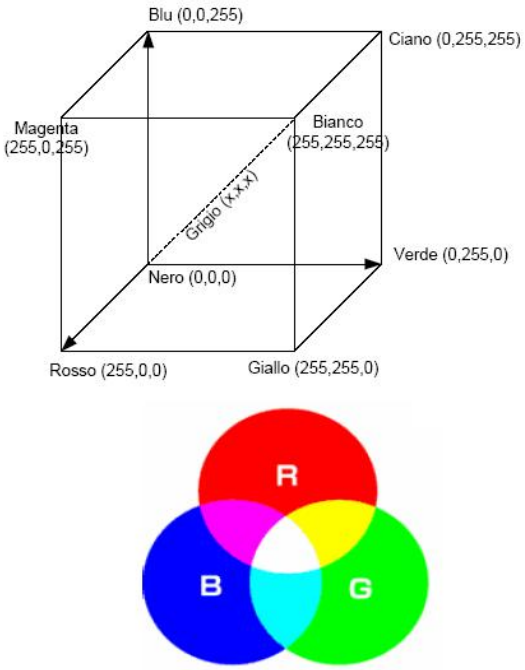


Figura 1: Spazio RGB

Pertanto si comprende facilmente che una qualsiasi immagine a colori può essere rappresentata da una funzione vettoriale come segue: X(m,n)=[x1(m,n), x2(m,n), x3(m,n)] in cui ciascuna componente è detta canale ed indica la funzione di luminosità in funzione delle coordinate spaziali di uno dei colori fondamentali (si veda la rappresentazione dello spazio RGB in figura sopra).

2.1 Classificazione Immagini

I principali tipi di immagine sono i seguenti:

1. Immagini a toni di grigio
2. Immagini a colori
3. Immagini binarie o in bianco e nero (B/W)

Un'immagine a toni di grigio è rappresentata come una matrice di interi indicanti i valori di intensità luminosa di ogni pixel su una scala di grigi. E' convenzione assumere i livelli discreti di grigio egualmente spaziati in un intervallo ben definito di valori, che normalmente va da 0 a 255 (nella seguente trattazione si utilizzano valori compresi tra 0 e 1). In genere il valore zero indica il nero, mentre il valore massimo indica il bianco. I valori compresi tra zero ed il valore massimo individuano tutti i toni di grigio intermedi. Questo tipo di immagine viene ampiamente utilizzato nella Computer Vision in quanto necessita di minore onerosità computazionale vista la trattazione di semplici matrici unidimensionali.

Per quanto riguarda le immagini a colori queste risultano più complesse da trattare vista la rappresentazione sia logica che fisica che hanno in memoria. Un colore infatti può allora

essere decomposto nella somma di tre colori fondamentali Rosso, Verde e Blu (RGB), ciascuno ovviamente preso con un'intensità opportuna. Dato che lo spazio RGB è uno spazio cartesiano, il colore di un pixel è rappresentato da un vettore e le sue componenti rappresentano la quantità di rosso, verde e blu rispettivamente necessarie a ottenere quel colore. Quindi un'immagine a colori può essere interpretata tramite una terna di matrici, ciascuna delle quali rappresenta un canale di colore: l'immagine nella sua interezza sarà allora la sovrapposizione delle immagini corrispondenti a ciascuna matrice ed equivalentemente ogni canale preso da solo è una immagine a toni di grigio e quindi può essere trattata di conseguenza. Le immagini digitali binarie ricoprono un ruolo di particolare importanza nelle applicazioni di Computer Vision in quanto alla base dei processi di rilevamento e riconoscimento di oggetti. Un'immagine è detta binaria se i suoi punti assumono solo due valori (tipicamente 0 e 1), ovvero se è costituita da soli due livelli di colore: bianco e nero. Si noti che si utilizza il termine punto indifferentemente da pixel in quanto le immagini digitali sono discretizzate, e quindi i punti sono rappresentati dai pixel.

2.2 Sistema di riferimento utilizzato per le immagini

Generalmente si fissa l'origine del sistema di riferimento in alto a sinistra dello schermo, con gli assi orientati come nella figura seguente:

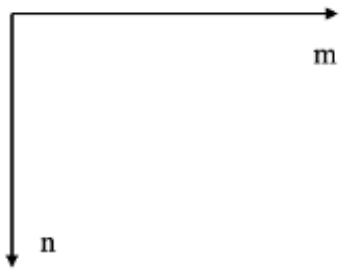


Figura 2: Sistema di Riferimento delle Immagini

La scelta di questo particolare sistema di riferimento risulta valido per tutti i tipi di immagini.

2.3 Immagini Binarie

Le immagini binarie sono immagini che vengono descritte su due valori, indicati in genere con 0 e 1, che rappresentano rispettivamente il nero e il bianco. Le immagini binarie sono utilizzate in molte applicazioni poiché sono più semplici da elaborare; tuttavia, essendo una rappresentazione impoverita delle informazioni dell'immagine originale, queste risultano utili dove tutte le informazioni necessarie possono essere fornite dalla sagoma dell'oggetto. Spesso il risultato di tecniche di elaborazione delle immagini è rappresentato sotto forma di un'immagine binaria, per esempio, il rilevamento dei bordi può essere un'immagine binaria. Tali tipi di immagini sono in genere ottenute tramite la sogliatura di un'immagine a livelli di grigio. I pixel con un livello di grigio al di sopra della soglia vengono impostati a 1, mentre gli altri vengono impostati a 0. Questo produce un oggetto bianco su fondo nero (o viceversa, a seconda dei valori relativi di grigio dell'oggetto e dello sfondo).

2.4 Filtri su Immagini Binarie

Nell'utilizzo di software di elaborazione delle immagini l'implementazione di operazioni quali smoothing o rilevamento dei contorni sono le attività principali che riguardano la Computer Vision. Queste sono solo alcune delle numerose operazioni realizzabili con dei filtri. La progettazione e l'utilizzo dei filtri è strettamente connessa allo studio di segnali e sistemi. In questo caso la grandezza da studiare è la quantità di colore di ogni pixel. E' facile intuire che si tratta di un segnale bidimensionale discreto e soprattutto che la variabile di riferimento è lo spazio: l'interesse riguarda infatti le variazioni spaziali del colore. Un sistema è un dispositivo che, dato un segnale in ingresso, lo elabora e ne fornisce il risultato in uscita. Un sistema è in genere una modellazione matematica di un fenomeno fisico. Sapendo la risposta impulsiva di un sistema e conoscendo l'espressione del segnale in ingresso, l'uscita del sistema può esser calcolata tramite l'operazione di convoluzione.
Operando con un segnale discreto e bidimensionale, l'immagine in uscita viene calcolata con la seguente operazione:

$$y(m,n) = x(m,n) * h(m,n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x(i,j) \cdot h(m-i, n-j)$$

dove M e N sono larghezza e altezza dell'immagine mentre il sistema è il filtro che si vuole applicare. Il filtro è un sistema che viene utilizzato solitamente per selezionare le frequenze di interesse ovvero il numero di variazioni di colore per unità spaziale. Come si può immaginare, sia il segnale che il sistema sono in realtà delle matrici bidimensionali, quindi l'operazione che stiamo per eseguire è una convoluzione fra matrici. La matrice che identifica il sistema è detta matrice o maschera di convoluzione o kernel ed è in genere una matrice quadrata di ordine dispari. La convoluzione fra matrici mira a calcolare il nuovo valore di ogni pixel sovrapponendo ad ognuno di essi la matrice kernel (con il centro sul punto in questione) ed eseguendo la sommatoria dei prodotti fra ogni pixel ed il corrispettivo elemento nella matrice di convoluzione.

3 Tecniche Edge-Based

Questo approccio sfrutta il fatto che gli oggetti che costituiscono un'immagine sono caratterizzati da valori di intensità molto diversi, per cui la capacità di rilevare le discontinuità equivale alla possibilità di individuare gli oggetti che la compongono. Un modo molto semplice per rilevare i bordi è attraverso un filtraggio spaziale in cui si assegnano opportunamente i coefficienti della maschera. Questa operazione è poi seguita da un processo di "Thresholding", che permette di produrre una mappa binaria in cui i contorni sono identificati solo dai pixel i cui valori di intensità superano una prefissata soglia T. La mappa così ottenuta spesso è caratterizzata da contorni piuttosto spessi, per cui tipicamente si applica una procedura per renderli più sottili, che prende il nome di "Thinning". In base a come si definisce la maschera è possibile individuare:

- punti isolati (Point Detection)
- linee (Line Detection)
- bordi di qualsiasi tipo (Edge Detection)

Nel seguito tratteremo separatamente queste elaborazioni.

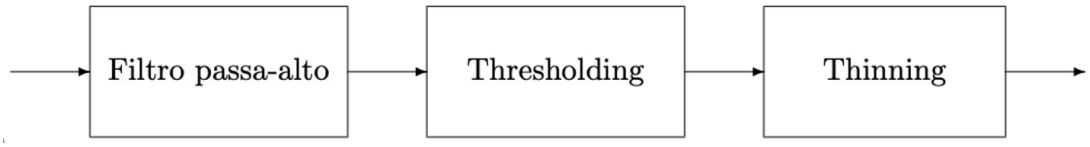


Figura 3: Diagramma a blocchi di un generico schema Edge Detection

3.1 Point Detection

Consideriamo il problema di rilevare punti isolati in un'immagine. Definiamo tali punti isolati come quei pixel il cui livello di grigio è significativamente diverso dallo sfondo, tipicamente omogeneo, in cui sono inseriti. In tal caso si può usare la seguente maschera:

$$h(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

L'idea che è alla base di questo approccio è che un punto isolato assumerà un valore molto diverso dai pixel circostanti, per cui quando la maschera è centrata proprio sul punto da rilevare produrrà una risposta molto grande; tale risposta sarà invece nulla in aree di intensità costante, dato che la somma di tutti i coefficienti della maschera è pari a zero.
Applichiamo questa procedura all'immagine di Figura 4, che mostra un'immagine a raggi X in cui è presente una porosità in alto a destra, dove si trova un singolo pixel nero. Scegliendo la soglia pari al 90% del livello di grigio più grande in valore assoluto, è possibile rilevare il punto isolato.

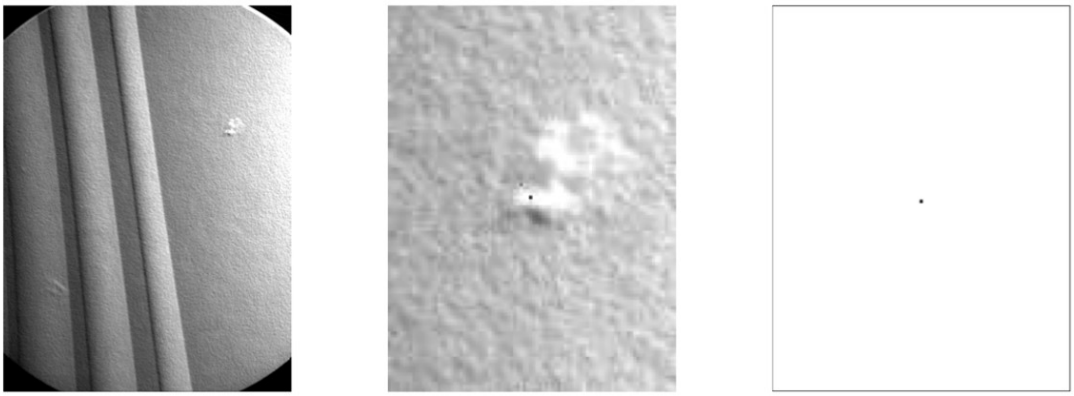


Figura 4: Immagine da elaborare, zoom della regione in cui è presente il punto isolato e mappa relativa

3.2 Line Detection

Per individuare le linee in un'immagine, usando la stessa strategia vista nel paragrafo precedente, bisogna modificare la maschera utilizzata. In particolare, si possono definire più maschere in base alla direzione che caratterizza la linea da estrarre:

$$h_1(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad h_2(m,n) = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

$$h_3(m,n) = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad h_4(m,n) = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Se l'immagine viene filtrata separatamente con queste 4 maschere si otterranno 4 immagini; $y_1(m,n)$, $y_2(m,n)$, $y_3(m,n)$ e $y_4(m,n)$, ognuna delle quali è caratterizzata da bordi nella direzione orizzontale, diagonale (+45°), verticale e diagonale (-45°), rispettivamente. Se per il punto generico relativo alla posizione m0, n0 della prima immagine $y_1(m_0,n_0)$ risulta:

$$|y_1(m_0,n_0)| > |y_i(m_0,n_0)| \;\; i = 2,3,4$$

allora il pixel in posizione (m_0,n_0) è associato ad una linea orizzontale. Si noti come la direzione che si vuole estrarre sia pesata con un coefficiente più grande rispetto alle altre direzioni. Per esempio, supponiamo di usare la prima maschera per la seguente sezione di immagine:

$$x(m,n) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 5 & 5 & 5 & 5 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

L'uscita presenta la massima risposta laddove sono presenti linee, di spessore pari a un pixel, orientate orizzontalmente:

$$y(m,n) = \begin{bmatrix} -6 & -9 & -9 & -9 & -6 \\ 16 & 24 & 24 & 24 & 16 \\ -6 & -9 & -9 & -9 & -6 \end{bmatrix}$$

Consideriamo come esempio l'immagine di figura 5, se siamo interessati ad individuare tutte le linee orientate a -45° bisognerà filtrare l'immagine usando l'ultima delle maschere indicate. Dallo zoom (Figura 6) si può notare quanto sia più forte la risposta del segmento obliquo in basso a destra piuttosto che quello in alto a sinistra e questo è legato al fatto che la componente in basso è spessa esattamente un pixel.

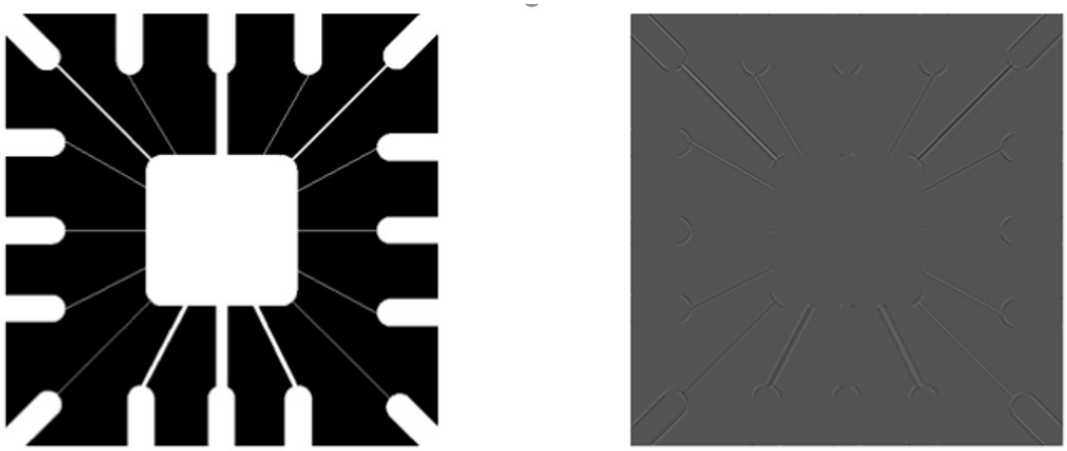


Figura 5: Immagine originale e filtrata con la maschera $h_4(m, n)$

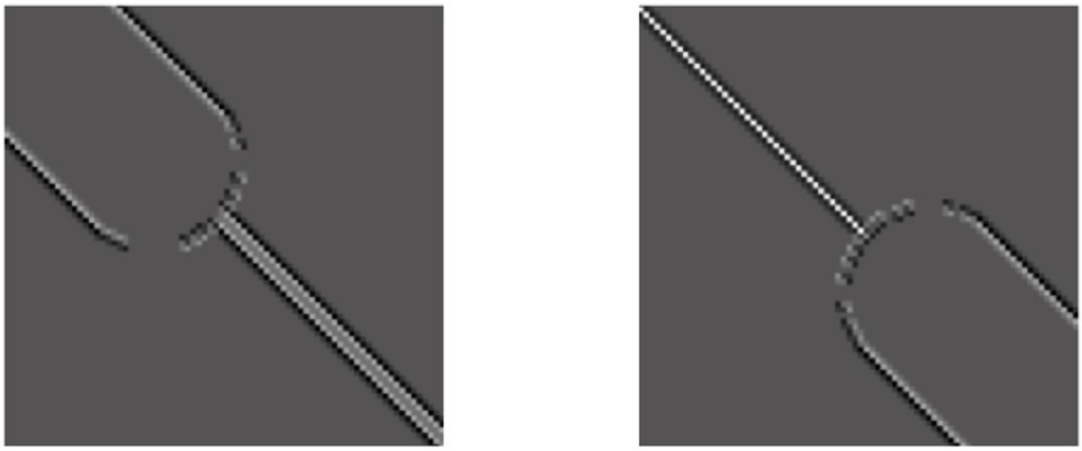


Figura 6: Zoom dell'immagine filtrata nelle regioni in cui sono presenti linee oblique

Se si vogliono determinare quali sono i punti che producono valori più elevati, basta effettuare ancora una volta un'operazione di Thresholding in cui la soglia è scelta come il valore massimo presente nell'immagine originale. In questo modo otteniamo un'immagine caratterizzata da linee di spessore pari proprio a un pixel orientate nella direzione -45°. In realtà nell'immagine (figura 7) sono anche presenti punti isolati, che possono essere rivelati con la tecnica descritta in precedenza e poi eliminati.

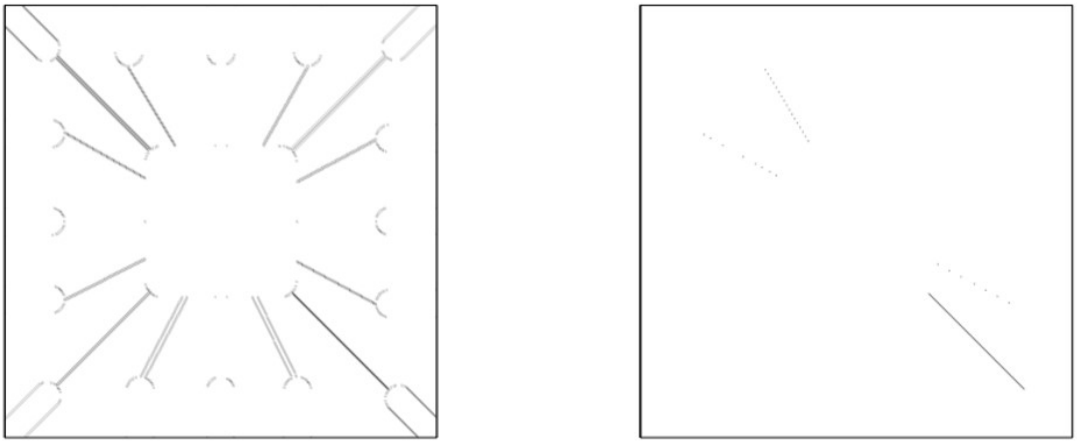


Figura 7: Valore assoluto dell'immagine filtrata e mappa binaria ottenuta dopo il Thresholding

3.3 Edge Detector

Sebbene il rilevamento di punti e linee sia molto importante nei processi di segmentazione, l'Edge Detection rappresenta di gran lunga l'approccio più comune per individuare le discontinuità nei livelli di grigio in un'immagine. Per affrontare e risolvere correttamente il problema è necessario definire un bordo. In figura 8 a sinistra è mostrata una regione in cui è presente una transizione ideale tra due livelli di luminosità e il corrispondente profilo di una sua sezione (Step Edge). Nella pratica, a causa di imprecisioni nell'acquisizione dell'immagine, il bordo non presenta una netta transizione tra i livelli di grigio, ma è caratterizzato da una variazione graduale che può essere ben approssimata da un modello a rampa (Figura 8 a destra).

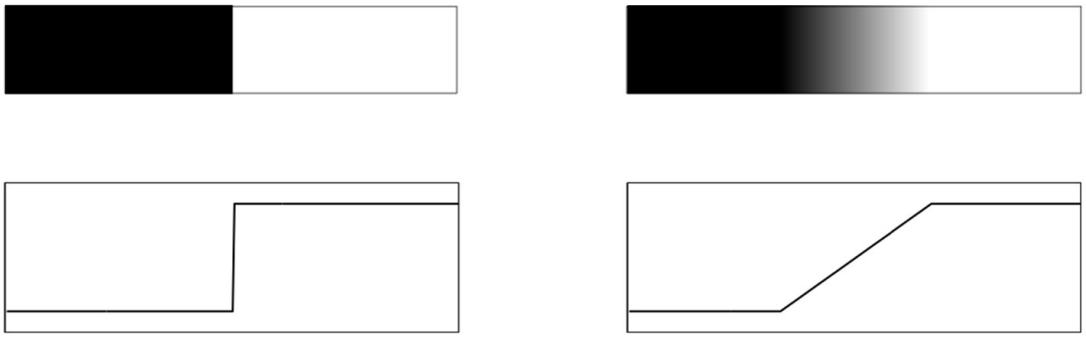


Figura 8: Bordo ideale e modello a rampa

Se il bordo presentasse una netta transizione, l'operazione che ne consentirebbe la rivelazione sarebbe la derivazione. In figura 9(a sinistra) si mostrano la derivata prima e la derivata seconda nel caso di edge a rampa: la derivata prima risulta costante per i punti appartenenti alla rampa, mentre è zero nelle aree costanti, quindi la sua ampiezza può essere usata per stabilire se un pixel dell'immagine è associato ad un bordo; la derivata seconda, invece, fornisce due valori in corrispondenza di ogni bordo. Nella pratica il profilo di un Edge a rampa è simile a quello mostrato in Figura 9 a destra. In tal caso la derivata prima assume valore massimo intorno al centro del bordo per cui dopo averla calcolata si può usare un'operazione di Thresholding per valutarne i massimi locali. La derivata seconda presenta, invece, l'interessante proprietà di attraversare lo zero intorno al punto centrale del bordo. Questa proprietà, detta di "zero-crossing", può risultare molto utile per localizzare il centro di bordi spessi.

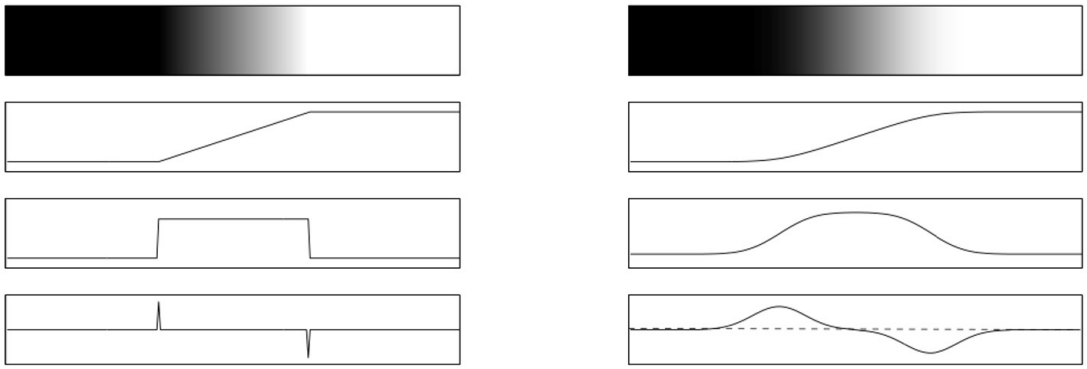


Figura 9: Derivata prima e seconda del bordo schematizzato con un modello a rampa

Spesso si ha a che fare con immagini rumorose, e anche se il rumore risulta quasi impercettibile nell'immagine, le operazioni di derivazione sono estremamente sensibili alla sua presenza. In Figura 10 si mostrano due esempi in cui alla sezione di immagine di Figura 9 è stato aggiunto un rumore gaussiano. Si può notare come, anche se dall'immagine e dal profilo difficilmente si riesca a percepire la presenza di rumore, la derivata prima e soprattutto la derivata seconda risultino fortemente distorte. Per questo motivo è necessario effettuare lo Smoothing delle immagini prima di applicare l'operazione di derivazione. Esiste però un Trade-Off tra la rilevazione corretta del bordo e l'individuazione della sua posizione. Infatti, per migliorare l'effetto del filtraggio del rumore, e quindi ridurre la probabilità di errata classificazione, è necessario aumentare la dimensione della maschera associata al filtro, d'altra parte in questo modo peggiora la capacità di localizzare perfettamente il bordo.

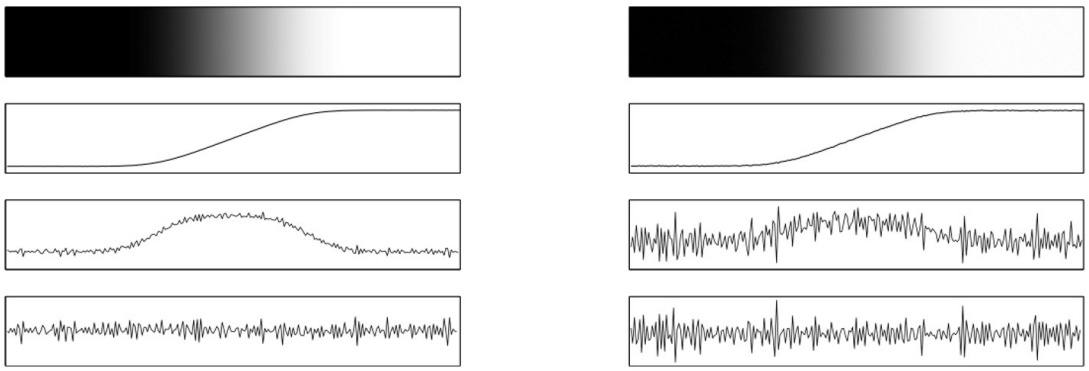


Figura 10: Esempi in cui è stato inserito rumore gaussiano

In conclusione per classificare un pixel dell'immagine come un bordo, la transizione su scala di grigi associata a quel punto deve essere molto più grande dello sfondo. Diremo allora che un pixel dell'immagine è un Edge Point se la derivata prima in quel punto è superiore ad una soglia fissata. Un insieme connesso di tali punti è un bordo (Edge). In alternativa possiamo definire gli Edge Point come i passaggi per lo zero della derivata seconda. E' bene tener presente che queste definizioni non garantiscono il successo nell'individuazione dei bordi in un'immagine. Il risultato in entrambi i casi è una mappa di contorni (Edge Map).

4 Smoothing

La diffusione è un processo fisico che riequilibra le differenze di concentrazione senza creare o distruggere la massa. Questa osservazione fisica può essere facilmente espressa in una formulazione matematica, la legge di Fick:

$$j = -D \cdot \nabla u$$

Questa equazione afferma che un gradiente di concentrazione ∇u provoca un flusso j che mira a compensare questo gradiente. La relazione tra ∇u e j è descritta dal tensore di diffusione D, una matrice simmetrica definita positiva. Il caso in cui j e ∇u sono paralleli è detto isotropo. Quindi possiamo sostituire il tensore di diffusione con una diffusività a valori scalari positivi c. Nel caso anisotropo generale, j e ∇u non sono paralleli.

L'osservazione che la diffusione trasporta solo massa senza distruggerla o creare nuova massa è espressa dall'equazione di continuità

$$\partial_t u = -divj$$

dove t indica il tempo.
Se inseriamo la legge di Fick nell'equazione di continuità, otteniamo l'equazione di diffusione:

$$\partial_t u = div(D \cdot \nabla u)$$

Questa equazione appare in molti processi di trasporto fisico. Nell'elaborazione delle immagini possiamo identificare la concentrazione con il valore di grigio in una certa posizione. Se il tensore

di diffusione è costante su tutto il dominio dell'immagine, si parla di diffusione omogenea, e un filtraggio spazio-dipendente è detto disomogeneo. Spesso il tensore di diffusione è una funzione della struttura differenziale dell'immagine in evoluzione stessa. Un tale feedback porta a filtri di diffusione non lineari. La diffusione che non dipende dall'immagine in evoluzione è chiamata lineare.

4.1 Processo di diffusione delle immagini al variare del tensore D

L'immagine varia nel tempo in accordo con la relazione

$$\partial_t u = div(D \cdot \nabla u)$$

Il tensore di diffusione D controlla tale processo. In particolare si distinguono i seguenti casi:

- Se D è uno scalare si ha la diffusione **Isotropa**;
- Se D è un tensore generico si ha la diffusione **Anisotropa**
- Se D è indipendente da u, si parla di diffusione **Lineare**
- Se D è dipendente da u, si parla di diffusione **Non Lineare**

Tutte le quattro combinazioni sono possibili, tuttavia nella seguente relazione tratteremo tali casi separatamente, concentrandoci in modo particolare, sul filtraggio a diffusione lineare. Tale tipo di filtraggio, infatti, sarà utilizzato per attuare lo Smoothing dell'immagine in input al Canny Edge Detector.

4.2 Filtraggio a diffusione lineare

Il miglior metodo a Partial Differential Equations (PDE) studiato per lo smoothing delle immagini consiste nell'applicare un processo di diffusione lineare. Ci concentreremo sulla relazione tra il filtraggio di diffusione lineare e la convoluzione con una gaussiana, analizzeremo le sue proprietà di smoothing per l'immagine e le sue derivate e passeremo in rassegna le proprietà fondamentali dello spazio-scala gaussiano indotto dal filtraggio di diffusione lineare. Successivamente forniremo un'indagine sugli aspetti discreti e discuteremo le applicazioni e le limitazioni del paradigma di diffusione lineare. La sezione si conclude abbozzando due generalizzazioni lineari che possono incorporare conoscenze a priori: spazio-scala gaussiano affine e processi di diffusione diretta.

Consideriamo il particolare caso della funzione lineare isotropa. In particolare il tensore di diffusione è una costante, ad esempio $D = c\ I$ dove I è la matrice unità. Dunque possiamo riscrivere l'equazione della diffusione come segue

$$\partial_t u = div(c \cdot \nabla u) = c \cdot \Delta u$$

dove Δu rappresenta l'operatore Laplaciano, ovvero :

$$\Delta u = div(\nabla u) = \frac{\partial_u^2}{\partial_x^2} + \frac{\partial_u^2}{\partial_y^2}$$

4.2.1 Relazioni con lo Smoothing Gaussiano

Sia un'immagine in scala di grigi f rappresentata da una mappatura a valori reali $f \in L^1(R^2)$.[1] Un modo ampiamente utilizzato per smussare l'immagine f è calcolare la convoluzione seguente:

$$(K_\sigma * f)(\mathbf{x}) := \int_{R^2} K_\sigma(\mathbf{x} - \mathbf{y}) f(\mathbf{y}) d\mathbf{y}$$

dove K_σ denota una funzione gaussiana bidimensionale della larghezza (deviazione standard) $\sigma > 0$:

$$K_\sigma(\mathbf{x}) := \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{|\mathbf{x}|^2}{2\sigma^2}}$$

Ci sono diversi motivi per le eccellenti proprietà di smussamento di questo metodo: innanzitutto osserviamo che poiché $K_\sigma \in C^\infty(R^2)$ otteniamo $K_\sigma * f \in C^\infty(R^2)$, anche se f è solo assolutamente integrabile.

Successivamente, esaminiamo il comportamento nel dominio della frequenza. Definiamo la Trasformata di Fourier F come segue:

$$(F\ f)(\omega) := \int_{R^2} f(x) e^{(-i<\omega,\mathbf{x}>)} dx$$

Otteniamo dal teorema di convoluzione che

$$(F(K_\sigma * f))(\omega) = (F\ K_\sigma)(\omega) \cdot (F\ f)(\omega)$$

Poiché la trasformata di Fourier di una gaussiana è di nuovo di forma gaussiana,

$$(F\ K_\sigma)(\omega) = e^{-\frac{|\omega|^2}{2/\sigma^2}}$$

osserviamo che la formula all'inizio del paragrafo è un filtro passa-basso che attenua le alte frequenze in modo monotono. È interessante notare che il comportamento levigante può anche essere compreso nel contesto di un'interpretazione PDE.

4.2.2 Equivalenza al filtraggio a diffusione lineare

Consideriamo il semplice caso in cui $c = 1$. Per ogni $f \in C$ intervallo limitato in R^2 il processo di diffusione lineare

$$\partial_t u = \Delta u$$
$$u(\mathbf{x}, 0) = f(\mathbf{x})$$

possiede la soluzione

[1] In analisi funzionale lo spazio L^p è lo spazio delle funzioni a p-esima potenza sommabile. Si tratta di uno spazio funzionale i cui elementi sono particolari classi di funzioni misurabili. In particolare si dice che una funzione $f \in L^1(R)$ se esiste finito l'integrale:

$$\int f(x)dx < \infty$$

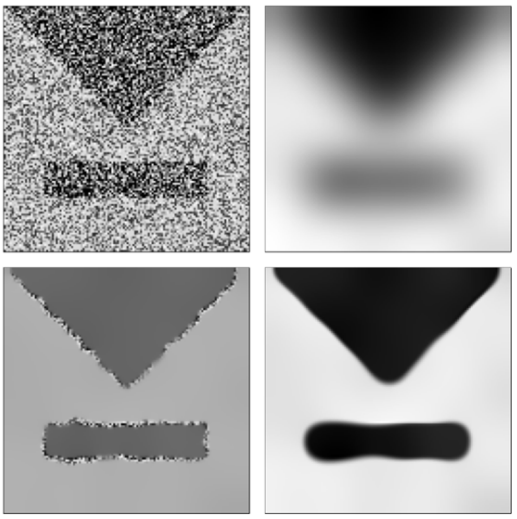
In generale $L^1(I)$ indica l'insieme delle funzioni che hanno integrale limitato sull'intervallo I. In generale le funzioni continue non sono integrabili su tutto R. Per esempio la funzione costante $f = 1$ non è $L^1(R)$ perché il suo integrale vale $+\infty$. Ovviamente il caso $f \in L^1(R^2)$ è l'estensione a due variabili del caso appena citato.

$$u(\mathbf{x},t) = \begin{cases} f(\mathbf{x}) & \text{per } t = 0 \\ \left(K_{\sqrt{2t}} * f\right)(\mathbf{x}) & \text{per } t \geq 0 \end{cases}$$

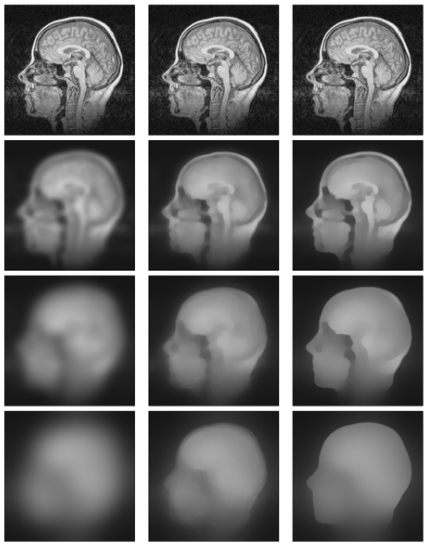
questa soluzione è unica.
Essa dipende continuamente dall'immagine iniziale f rispetto a $\|\cdot\|_{L^\infty(R^2)}$ e soddisfa il principio del massimo-minimo

$$inf_{R^2} f \leq u(\mathbf{x},t) \leq sup_{R^2}\ f \quad \text{su } R^2 \times [0,\infty)$$

Dalla soluzione $u(\mathbf{x},t)$ osserviamo che il tempo t è correlato all'ampiezza spaziale $\sigma = \sqrt{2t}$ della Gaussiana. Quindi, il livellamento delle strutture di ordine σ richiede di fermare il processo di diffusione al tempo $T = \frac{\sigma^2}{2}$.



Restoration properties of diffusion filters. (a) Top Left: Test image, $\Omega = (0, 128)^2$. (b) Top Right: Linear diffusion, $t = 80$. (c) Bottom Left: Nonlinear isotropic diffusion, $\lambda = 3.5$, $\sigma = 3$, $t = 80$. (d) Bottom Right: Nonlinear anisotropic diffusion, $\lambda = 3.5$, $\sigma = 3$, $t = 80$.



Evolution of an MRI slice under different PDEs. Top: Original image, $\Omega = (0, 236)^2$. (a) Left Column: Linear diffusion, top to bottom: $t = 0$, 12.5, 50, 200. (b) Middle Column: Inhomogeneous linear diffusion ($\lambda = 8$), $t = 0$, 70, 200, 600. (c) Right Column: Nonlinear isotropic diffusion with the Charbonnier diffusivity ($\lambda = 3$), $t = 0$, 70, 150, 400.

Figura 11: Effetto del filtraggio a diffusione lineare

4.2.3 Proprietà della rappresentazione spazio-scala

È un fatto ben noto che le immagini di solito contengono strutture a una grande varietà di scale. In quei casi in cui non è chiaro in anticipo quale sia la scala giusta per le informazioni rappresentate, è desiderabile avere una rappresentazione dell'immagine su più scale. Inoltre, confrontando le strutture a diverse scale, si ottiene una gerarchia di strutture di immagine che facilita una successiva interpretazione dell'immagine. Il motivo per generare una rappresentazione spazio-scala di un dato set di dati deriva dall'osservazione di base che gli oggetti del mondo reale sono composti da strutture diverse a scale diverse. Ciò implica che gli oggetti del mondo reale, in contrasto con entità matematiche idealizzate come punti o linee, possono apparire in modi diversi a seconda della scala di osservazione. Ad esempio, il concetto di "albero" è appropriato alla scala dei metri, mentre concetti come foglie e molecole sono più appropriati a scale più fini.

Per un sistema di visione artificiale che analizza una scena sconosciuta, non c'è modo di sapere a priori quali scale sono appropriate per descrivere le strutture interessanti nei dati dell'immagine. Quindi, l'unico approccio ragionevole è considerare descrizioni su più scale per poter catturare le variazioni di scala sconosciute che possono verificarsi. Portata al limite, una rappresentazione spazio-scala considera rappresentazioni a tutte le scale. Un'altra motivazione al concetto di spazio scala deriva dal processo di esecuzione di una misurazione fisica su dati del mondo reale. Per estrarre qualsiasi informazione da un processo di misurazione, è necessario applicare ai dati operatori di dimensione non infinitesimale. In molti rami dell'informatica e della matematica applicata, la dimensione dell'operatore di misura è trascurata nella modellazione teorica di un problema. La teoria dello spazio scala, d'altra parte, incorpora esplicitamente la necessità di una dimensione non infinitesimale degli operatori immagine come parte integrante di qualsiasi misurazione e di qualsiasi altra operazione che dipenda da una misurazione del mondo reale.

Uno spazio-scala è una rappresentazione dell'immagine in un *continuum* di scale, incorporando l'immagine f in una famiglia $\{L_t f | t \geq 0\}$ di versioni gradualmente semplificate di essa, a condizione che soddisfi determinati requisiti. La maggior parte di queste proprietà può essere classificata come requisiti architetturali, di livellamento (riduzione delle informazioni) o di invarianza.

Un'importante ipotesi architettonica è la ricorsività, cioè per $t = 0$, la rappresentazione dello spazio scala fornisce l'immagine originale f, e il filtraggio può essere suddiviso in una sequenza di banchi di filtri:

$$L_0 f = f$$

$$L_{t+s}\ f = L_t(L_s f)\ \ \forall s, t \geq 0$$

Questa proprietà è molto spesso indicata come **proprietà di semi-gruppo**. Altri principi architettonici comprendono, ad esempio, le proprietà di regolarità di L_t e il comportamento locale quando t tende a 0. Le proprietà leviganti e la riduzione delle informazioni nascono dal desiderio che la trasformazione non crei alterazioni quando si passa dalla rappresentazione fine a quella grossolana. Pertanto, su scala grossolana, non dovremmo avere strutture aggiuntive che sono causate dal metodo di filtraggio stesso e non da strutture sottostanti su scale più fini.

Possiamo considerare un'immagine come rappresentazione di una classe di equivalenza contenente tutte le immagini che raffigurano lo stesso oggetto. Due immagini di questa classe differiscono, ad esempio da spostamenti di livello di grigio, traslazioni e rotazioni o trasformazioni anche più complicate come le mappature affini. Ciò rende plausibile l'esigenza che l'analisi dello spazio scala sia invariante al maggior numero possibile di queste trasformazioni, al fine di analizzare solo l'oggetto rappresentato. Le PDE sono la struttura adatta per gli spazi di scala e spesso sono integrate da un'ipotesi aggiuntiva che è equivalente al principio di sovrapposizione, ovvero la linearità:

$$L_t(af + bg) = aL_t f + bL_t g \quad \forall t \geq 0,\ \ \forall a, b \in R$$

La nozione di spazio scala si applica ai segnali di un numero arbitrario di variabili. In particolare si applica alle immagini bidimensionali: data una immagine $f(x, y)$, la sua rappresentazione in scala lineare (gaussiana) è una famiglia di segnali derivati $L(x, y; t)$ definita dalla convoluzione di $f(x, y)$ con un kernel bidimensionale gaussiano:

$$K_{\sqrt{2t}}(x, y; t) = \frac{1}{4\pi t} \cdot e^{-\frac{x^2+y^2}{4t}}$$

$$L(\cdot\ ,\ \cdot\ ;\ t) = K_\sigma(\cdot\ ,\ \cdot\ ;\ t) * f(\cdot\ ,\ \cdot)$$

dove il punto e virgola nell'argomento di L implica che la convoluzione viene eseguita solo sulle variabili x,y, mentre il parametro di scala t (dopo il punto e virgola) indica solo quale livello di scala si sta definendo. Questa definizione di L funziona per un *continuum* di scale $t \geq 0$, ma tipicamente verrebbe considerato solo un insieme discreto finito di livelli nella rappresentazione dello spazio di scala. Il parametro di scala $t = \sigma^2/2$ è la varianza del filtro gaussiano. Per $t = 0$ il filtro K diventa una funzione impulsiva tale che $L(x, y; 0) = f(x, y)$, ovvero la rappresentazione spazio-scala in questo caso è l'immagine f stessa. All'aumentare di t, L è il risultato dell'attenuazione di f con un filtro di dimensione sempre maggiore, che, quindi, rimuoverà un maggior numero di dettagli dell'immagine. Poiché la deviazione standard del filtro è $\sigma = \sqrt{2t}$, i dettagli che sono significativamente inferiori a questo valore vengono in gran parte rimossi dall'immagine al parametro di scala t.

Figura 12: Esempio rappresentazione Spazio-Scala

4.2.4 Spazio-Scala Gaussiano

Il primo e miglior studiato spazio scala è lo spazio scala gaussiano, che si ottiene per convoluzione con gaussiane a varianza crescente, o - equivalentemente - per filtraggio per diffusione lineare secondo

$$\partial_t u = \Delta u$$
$$u(\mathbf{x}, 0) = f(\mathbf{x})$$

Perché un filtro gaussiano?
Di fronte al compito di generare una rappresentazione multi-scala ci si può chiedere: per generare uno spazio di scala si può utilizzare un qualsiasi filtro K di tipo passa basso con un parametro t che ne determina la larghezza? La risposta è no, poiché è di cruciale importanza che il filtro di livellamento non introduca nuove strutture spurie a scale grossolane che non corrispondano a semplificazioni di strutture corrispondenti a scale più fini. Nella letteratura dello spazio scala, sono stati espressi diversi modi per formulare questo criterio in termini matematici precisi. La conclusione da diverse derivazioni assiomatiche che sono state presentate è che lo spazio della scala gaussiana costituisce il modo canonico per generare uno spazio-scala lineare, sulla base del requisito essenziale che non si devono creare nuove strutture quando si passa da una scala fine a una scala più grossolana. Le condizioni, denominate **assiomi spazio-scala**, che sono state utilizzate per derivare l'unicità del kernel gaussiano includono linearità, invarianza di traslazione, proprietà a semi-gruppo, invarianza di scala e conservazione della positività.

Una derivazione assiomatica dello spazio-scala gaussiano 1-D considera una trasformazione di osservazione Φ che dipende da un parametro di scala σ e che trasforma l'immagine originale f($\mathbf{x}$) in una versione smoothed $\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma]$ Questa classe di trasformazioni di sfocatura è chiamata *boke* (defocusing).
Si presume che si abbia la struttura seguente:

$$\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma] = \int_{-\infty}^{+\infty} \phi\{f(\mathbf{x}'), \mathbf{x}, \mathbf{x}', \sigma\} d\mathbf{x}'$$

e che dovrebbe soddisfare cinque condizioni:

1. **Linearità (rispetto alle moltiplicazioni):**
   Se l'intensità di un pattern diventa A volte la sua intensità originale, lo stesso dovrebbe accadere al pattern osservato:

   $$\Phi[Af(\mathbf{x}'), \mathbf{x}, \sigma] = A\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma]$$

2. **Invarianza di traslazione:**
   Filtrare un'immagine traslata equivale a traslare l'immagine filtrata:

   $$\Phi[f(\mathbf{x}' - a), \mathbf{x}, \sigma] = \Phi[f(\mathbf{x}'), \mathbf{x} - a, \sigma]$$

3. **Invarianza di Scala:**
   Se un pattern è allargato spazialmente di qualche fattore λ, allora esiste un $\sigma' = \sigma'(\sigma, \lambda)$ tale che:

   $$\Phi[f(\mathbf{x}'/\lambda), \mathbf{x}, \sigma] = \Phi[f(\mathbf{x}'), \mathbf{x}/\lambda, \sigma]$$

4. **Proprietà di semi-gruppo(generalizzata):**
   Se f è osservato sotto un parametro σ_1 e questa trasformazione è osservata sotto un parametro σ_2, allora questo è equivalente ad osservare f sotto un parametro appropriato $\sigma_3 = \sigma_3(\sigma_1, \sigma_2)$:
   $$\Phi[\Phi[f(\mathbf{x}''), \mathbf{x}', \sigma_1], \mathbf{x}, \sigma_2] = \Phi[f(\mathbf{x}''), \mathbf{x}, \sigma_3]$$

5. **Conservazione della positività:**
   Se l'immagine originale è positiva, anche l'immagine osservata è positiva:
   $$\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma] > 0 \;\; \forall f(\mathbf{x}') > 0, \;\; \forall \sigma > 0$$

Sotto questi requisiti si deriva in modo molto sistematico che:

$$\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma] = \frac{1}{2\sqrt{\pi}\sigma} \cdot \int_{-\infty}^{+\infty} f(\mathbf{x}') \cdot e^{-\frac{(\mathbf{x}-\mathbf{x}')^2}{4\sigma^2}} d\mathbf{x}'$$

Quindi, $\Phi[f(\mathbf{x}'), \mathbf{x}, \sigma]$ è solo la convoluzione tra f e una gaussiana con deviazione standard $\sigma\sqrt{2}$.

Equivalentemente, come già illustrato precedentemente, la famiglia spazio-scala può essere definita come la soluzione dell'equazione di diffusione,

$$\partial_t L = \Delta L$$

con condizione iniziale $L(x, y; 0) = f(x, y)$. Questa formulazione della rappresentazione spazio-scala L significa che è possibile interpretare i valori dell'intensità dell'immagine f come una "distribuzione di temperatura" nel piano dell'immagine e che il processo che genera la rappresentazione spazio-scala in funzione di t corrisponda alla diffusione del calore nel piano dell'immagine nel tempo t. Sebbene questa connessione possa apparire superficiale per un lettore che non ha familiarità con le equazioni differenziali, è in effetti il caso che la principale formulazione spazio-scala in termini di non miglioramento degli estremi locali è espressa in termini di una condizione di segno sulle derivate parziali nel volume di dimensione (2+1)-D generato dallo spazio scala, quindi nell'ambito di equazioni differenziali alle derivate parziali. Inoltre, un'analisi dettagliata del caso discreto mostra che l'equazione di diffusione fornisce un collegamento unificante tra spazi a scala continua e discreta, che generalizza anche a spazi a scala non lineare, ad esempio, utilizzando la diffusione anisotropa. Quindi, si può dire che il modo principale per generare uno spazio di scala è l'equazione di diffusione, e che il kernel gaussiano nasce come funzione di Green di questa specifica equazione differenziale parziale.

4.2.5 Derivate Gaussiane

Per comprendere la struttura di un'immagine dobbiamo analizzare le variazioni del valore del grigio nell'intorno di ciascun punto dell'immagine, vale a dire, abbiamo bisogno di informazioni sulle sue derivate. Tuttavia, la differenziazione è mal posta, poiché piccole perturbazioni nell'immagine originale possono portare a variazioni arbitrariamente grandi nelle derivate. Da qui nasce la necessità di metodi di regolarizzazione. Una possibilità per regolarizzare l'immagine è convolvere l'immagine con una gaussiana prima di derivarla. Dall'uguaglianza:

$$\partial_x^n \partial_y^m (K_\sigma * f) = K_\sigma * (\partial_x^n \partial_y^m \; f) = (\partial_x^n \partial_y^m \; K_\sigma) * f$$

per f sufficientemente liscia, osserviamo che tutte le derivate subiscono lo stesso processo di levigatura gaussiana dell'immagine stessa e questo processo è equivalente a convolvere l'immagine con le derivate di una gaussiana. La sostituzione delle derivate con queste derivate gaussiane ha un forte effetto regolarizzante. Inoltre, le derivate gaussiane possono essere combinate con i cosiddetti invarianti differenziali, espressioni invarianti per trasformazioni come le rotazioni, ad esempio $|\nabla K_\sigma * u|$ o $\Delta K_\sigma * u$.
Riassumendo, a qualsiasi scala nello spazio-scala, possiamo applicare operatori derivativi locali alla rappresentazione dello spazio-scala. A causa della proprietà commutativa tra l'operatore derivato e l'operatore di livellamento gaussiano, tali derivate spazio-scala possono essere calcolate in modo equivalente convolvendo l'immagine originale con operatori derivativi gaussiani. Per questo motivo sono spesso indicati anche come derivati gaussiani:

$$(\partial_x^n \partial_y^m L)(x, y; t) = \partial_x^n \partial_y^m K_t(x, y; t) * f(x, y)$$

Gli invarianti differenziali sono utili per il rilevamento di caratteristiche quali bordi, creste, giunzioni e blob. Un metodo usato frequentemente per il rilevamento dei bordi è il Canny Edge Detector. Si basa sul calcolo delle prime derivate dell'immagine gaussiana levigata. Dopo aver applicato sofisticati meccanismi di assottigliamento e collegamento (soppressione non massima e soglia di isteresi), i bordi vengono identificati come posizioni in cui l'ampiezza del gradiente ha un massimo. Questo metodo è spesso riconosciuto come il miglior rilevatore di bordi lineari ed è diventato uno standard nel rilevamento dei bordi.
Se si indaga l'evoluzione temporale dei passaggi per lo zero di un'immagine filtrata per diffusione lineare, si osserva un fenomeno interessante: quando si aumenta la scala di livellamento σ, non si creano nuovi passaggi per lo zero che non possano essere ricondotti a scale più fini. Questa proprietà dell'evoluzione è chiamata *causalità*. La proprietà evolutiva degli zero-crossing è stata l'indagine chiave che ha ispirato il cosiddetto concetto di scala-spazio. Questo sarà discusso in seguito.

4.2.6 Aspetti Numerici

La teoria precedente è interamente nel continuo. Tuttavia, in problemi pratici, l'immagine viene campionata ai nodi (pixel) di una griglia equidistante fissa. Pertanto, il filtro di diffusione deve essere discretizzato. In virtù dell'equivalenza tra la risoluzione dell'equazione di diffusione lineare e la convoluzione con una gaussiana, possiamo approssimare il processo di convoluzione o l'equazione di diffusione. Quando si restringe l'immagine a un dominio finito e si applica la Fast Fourier Transformation (FFT), la convoluzione nel dominio spaziale può essere ridotta alla moltiplicazione nel dominio della frequenza. Questo procedimento richiede uno sforzo computazionale fisso di ordine $N \cdot logN$, che dipende solo dal numero di pixel N, ma non dalla dimensione del kernel σ. Per i kernel di grandi dimensioni questo è più veloce della maggior parte delle tecniche spaziali. Soprattutto per i kernel piccoli, tuttavia, gli effetti di aliasing nel dominio di Fourier possono creare oscillazioni. Una possibilità efficiente per approssimare la convoluzione gaussiana nel dominio spaziale consiste nell'applicare filtri ricorsivi. Più frequentemente il kernel gaussiano viene semplicemente campionato e troncato a qualche multiplo della sua deviazione standard. La fattorizzazione di una gaussiana di dimensione superiore in gaussiane unidimensionali riduce lo sforzo computazionale a $O(N\sigma)$. La convoluzione con una gaussiana troncata, tuttavia, rivela l'inconveniente di non preservare la proprietà del semi-gruppo dello spazio di

scala gaussiano continuo. Tra le numerose possibilità numeriche per approssimare l'equazione di diffusione lineare dominano gli schemi alle Differenze Finite (FD). Tra di essi vengono utilizzati principalmente schemi espliciti. Un'approssimazione molto efficiente dello spazio scala gaussiano risulta dall'applicazione di multi-griglie. La piramide gaussiana ha la complessità computazionale $O(N)$ e fornisce una rappresentazione multi-livello a un numero finito di scale di diversa risoluzione. Smussando successivamente l'immagine con uno schema esplicito per l'equazione di diffusione e limitando il risultato a una griglia più grossolana, si ottiene una rappresentazione dell'immagine semplificata alla griglia successiva più grossolana. A causa della loro semplicità ed efficienza, le scomposizioni piramidali sono diventate molto utilizzate

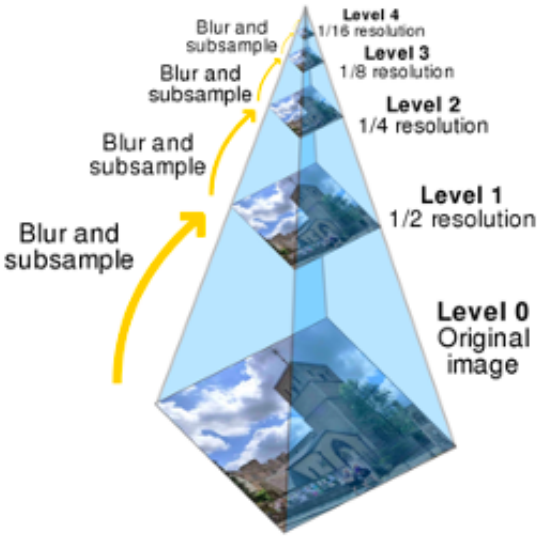


Figura 13: Piramide Gaussiana

4.2.7 Limitazioni

Nonostante le numerose proprietà che rendono il filtraggio a diffusione lineare unico e maneggevole, presenta anche alcuni inconvenienti:

1. Un ovvio svantaggio del livellamento gaussiano è il fatto che non solo attenua il rumore, ma offusca anche caratteristiche importanti come i bordi e, quindi, li rende più difficili da identificare.

2. Il filtraggio a diffusione lineare disloca i bordi quando si passa da scale più fini a scale più grossolane. Quindi le strutture che sono identificate su scala grossolana non danno la giusta posizione e devono essere ricondotte all'immagine originale.

3. Alcune proprietà di levigatura dello spazio scala gaussiano non vengono trasferite dal caso 1-D a dimensioni superiori.

4.2.8 Generalizzazioni

Una piramide di immagini è una rappresentazione discreta in cui uno spazio in scala viene campionato sia nello spazio che nella scala. Per l'invarianza di scala, i fattori di scala dovrebbero essere campionati in modo esponenziale, ad esempio come potenze intere di 2 o $\sqrt{2}$. Quando costruito correttamente, il rapporto tra le frequenze di campionamento nello spazio e la scala sono mantenuti costanti in modo che la risposta all'impulso sia identica in tutti i livelli della piramide. Esistono algoritmi veloci, per calcolare una piramide di immagini invarianti di scala, in cui l'immagine o il segnale viene ripetutamente livellato e poi sotto-campionato. I valori per lo spazio di scala tra i campioni piramidali possono essere facilmente stimati utilizzando l'interpolazione all'interno e tra le scale e consentendo stime di scala e posizione con precisione di risoluzione inferiore. In una rappresentazione spazio-scala, l'esistenza di un parametro di scala continuo consente di tracciare passaggi per lo zero su scale che portano alla cosiddetta struttura profonda. Le estensioni della teoria dello spazio scala lineare riguardano la formulazione di concetti spazio scala non lineari. Questi partono spesso dalla formulazione di diffusione equivalente del concetto di spazio di scala, che viene successivamente esteso in modo non lineare. In questo modo sono state formulate un gran numero di equazioni di evoluzione, motivate da diverse esigenze specifiche (si veda la bibliografia sopra citata per maggiori informazioni). Va notato, tuttavia, che non tutti questi spazi di scala non lineari soddisfano requisiti teorici simili a quelli del concetto di spazio di scala gaussiano lineare. Un'estensione del primo ordine dello spazio della scala gaussiana isotropa è fornita dallo spazio della scala affine (gaussiana). Una motivazione per questa estensione deriva dalla necessità comune di calcolare descrittori di immagini soggetti a oggetti del mondo reale che vengono visualizzati sotto un modello di telecamera prospettica. Per gestire localmente tali deformazioni non lineari, l'invarianza parziale alle deformazioni affini locali può essere ottenuta considerando kernel gaussiani affini con le loro forme determinate dalla struttura dell'immagine locale. Infatti, questo spazio di scala affine può anche essere espresso da un'estensione non isotropa dell'equazione di diffusione lineare (isotropa), pur essendo ancora all'interno della classe delle equazioni alle derivate parziali lineari. Oltre alle variabilità su scala, per le quali la teoria originale dello spazio scala è stata progettata, questa teoria dello spazio scala generalizzata comprende anche altri tipi di variabilità causati da trasformazioni geometriche nel processo di formazione dell'immagine, comprese le variazioni nella direzione di visualizzazione approssimata da trasformazioni affini locali e moti relativi tra gli oggetti nel mondo e l'osservatore, approssimati da trasformazioni galileiane locali.

4.2.9 Spazio-scala gaussiano affine

Una semplice generalizzazione dello spazio-scala gaussiano risulta dalla rinuncia all'invarianza sotto le rotazioni. Questo porta allo spazio scala gaussiano affine

$$u(\mathbf{x}, t) := \int_{R^2} \frac{1}{4\pi\sqrt{det(D_t)}} \cdot exp\left(-\frac{(\mathbf{x}-\mathbf{y})^T\ D_t^{-1}\ (\mathbf{x}-\mathbf{y})}{4}\right) f(\mathbf{y}) d\mathbf{y}$$

dove $D_t := tD, t > 0$, e $D \in R^{2\times 2}$ è simmetrica definita positiva. Per una matrice fissa (fixed) D, calcolare l'integrale di convoluzione (formula riportata sopra) equivale a risolvere un problema di diffusione anisotropa lineare con D come tensore di diffusione:

$$\partial_t u = div(D\nabla u)$$

$$u(\mathbf{x}, 0) = f(\mathbf{x})$$

Definendo $u(\mathbf{x}, t)$ la figura generalizzata di f. Va notato che il livellamento gaussiano adattato alla forma con una D variabile nello spazio non è più equivalente a un processo di diffusione del tipo scritto in precedenza. In pratica ciò può essere sperimentato dal fatto che l'adattamento di forma della levigatura gaussiana non preserva il livello medio di grigio, mentre la formulazione della divergenza assicura che ciò sia ancora possibile per il filtraggio a diffusione non uniforme. Anche in questo caso l'equazione della diffusione sembra essere più generale. Se si vuole mettere in relazione il livellamento gaussiano adattato alla forma con una PDE, è necessario applicare limiti di scala sofisticati. Il livellamento gaussiano non iterativo adattato alla forma differisce dal filtraggio anisotropico non lineare per diffusione per il fatto che quest'ultimo introduce un feedback nel processo: adatta il tensore di diffusione alla struttura differenziale dell'immagine filtrata invece dell'immagine originale.

4.2.10 Diffusione diretta

Un altro metodo per incorporare la conoscenza a priori in un processo di diffusione lineare è la diffusione diretta. A condizione che ci vengano fornite alcune informazioni di base sotto forma di un'immagine liscia b, si ha che a condizioni al contorno adeguate la soluzione classica u di

$$\partial_t u = b\Delta u - u\Delta b$$

$$u(x, 0) = f(x)$$

converge a b lungo un percorso in cui l'entropia relativa rispetto a b aumenta in modo monotono. Un tale processo di diffusione diretta richiede di specificare in anticipo un'intera immagine come informazione di sfondo; in molte applicazioni sarebbe desiderabile includere la conoscenza a priori in un modo meno specifico, ad esempio prescrivendo che le caratteristiche all'interno di un certo contrasto e intervallo di scala siano considerate semanticamente importanti ed elaborate in modo diverso. Tali richieste possono essere soddisfatte da filtri di diffusione non lineare.

4.3 Filtraggio a diffusione non lineare

I metodi di Smoothing si basano sull'idea di applicare un processo che a sua volta dipende dalle proprietà locali dell'immagine. Queste tecniche possono essere estese a processi anisotropi che utilizzano un tensore di diffusione adattato invece di una diffusività scalare.

$$c \cdot \Delta u \equiv c(x, y, I) \cdot \Delta u$$

4.3.1 Modello di Perona-Malik

Perona e Malik propongono un metodo di diffusione non lineare per evitare i problemi di sfocatura e localizzazione del filtraggio a diffusione lineare, applicando un processo disomogeneo che riduce la diffusività in quei punti che hanno una maggiore probabilità di essere bordi. Questa probabilità è misurata da $|\nabla u|^2$. Il filtro Perona– Malik si basa sull'equazione

$$\partial_t u = div\left(g(|\nabla u|^2)\nabla u\right)$$

e utilizza diffusività come

$$g(s^2) = \frac{1}{1 + s^2/\lambda^2} \quad (\lambda > 0)$$

Il modello è isotropo, poiché utilizza una diffusività a valori scalari e non un tensore di diffusione. Gli esperimenti di Perona e Malik sono stati visivamente molto impressionanti: i bordi rimangono stabili per un tempo molto lungo. Il rilevamento dei bordi basato su questo processo supera chiaramente del Canny Edge Detector (lineare), anche senza applicare la soppressione non massima e la soglia di isteresi. Ciò è dovuto al fatto che la diffusione e il rilevamento dei bordi interagiscono in un unico processo invece di essere trattati come due processi indipendenti che devono essere applicati successivamente.

4.3.2 Miglioramento dei bordi

Per studiare il comportamento del filtro Perona–Malik ai bordi, limitiamoci per un momento al caso unidimensionale. Ciò semplifica la notazione e illustra il comportamento principale poiché vicino a un bordo rettilineo un'immagine bidimensionale approssima una funzione di una variabile.
Per la diffusività segue che la *funzione di flusso* $\Phi(s) := sg(s^2)$ soddisfa $\Phi'(s) \geq 0$ per $|s| \leq \lambda$, e $\Phi'(s) \leq 0$ per $|s| > \lambda$. Poiché l'equazione su cui si basa il filtro di Perona-Malik può essere riscritta come:

$$\partial_t u = \Phi'(u_x) u_{xx}$$

osserviamo che, nonostante la sua diffusività non negativa, il modello di Perona–Malik è di tipo parabolico in avanti per $|ux| \leq \lambda$, e di tipo parabolico all'indietro per $|ux| > \lambda$.

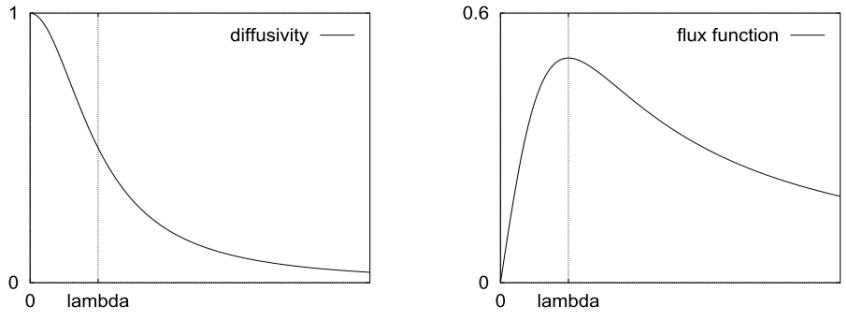


Figura 14: Diffusività (SX) e Funzione di flusso (DX)

Quindi, λ svolge il ruolo di un parametro di contrasto che separa le aree di diffusione in avanti (basso contrasto) da quelle indietro (alto contrasto). Non è difficile verificare che il filtro Perona–Malik aumenti la pendenza nei punti di flesso dei bordi all'interno di un'area arretrata: se esiste una soluzione sufficientemente liscia u soddisfa

$$\partial_t(u_x^2) = 2u_x\partial_x(u_t) = 2\Phi''(u_x)u_x u_{xx}^2 + 2\Phi'(u_x)u_x u_{xxx}$$

Una posizione x_0 dove u_x^2 è massima in un certo momento t è caratterizzata da $u_x u_{xx} = 0$ e $u_x u_{xxx} \leq 0$. Perciò:

$$(\partial_t(u_x^2))(x_0, t) \geq 0 \text{ per } |u_x(x_0, t)| > \lambda$$

Con stretta disuguaglianza per $u_x u_{xxx} < 0$. Nel caso bidimensionale,l'equazione su cui è basato il filtro diventa:

$$\partial_t u = \Phi'(\nabla u) u_{\eta\eta} + g(|\nabla u|^2) u_{\xi\xi}$$

dove le *coordinate di gauge* ξ e η indicano le direzioni perpendicolari e parallele a ∇u, rispettivamente. Quindi, abbiamo diffusione in avanti lungo le isofotiche (cioè linee di valore di grigio costante) combinata con diffusione avanti-indietro lungo le linee di flusso (linee di variazione del valore di grigio massimo). Osserviamo che il comportamento di diffusione avanti-indietro non è limitato solo alla diffusività speciale, appare per tutte le diffusività $g(s^2)$ il cui rapido decadimento causa funzioni di flusso non monotone $\Phi(s) = s\ g(s^2)$. Le diffusività in rapida diminuzione sono esplicitamente intese nel metodo Perona – Malik poiché danno il risultato desiderabile di sfocare piccole fluttuazioni e affilare i bordi. Pertanto, sono la ragione principale dei risultati visivamente impressionanti di questa tecnica di restauro.

4.4 Schema numerico per la risoluzione dell'equazione di diffusione

Come ricavato precedentemente, l'equazione di diffusione lineare omogenea con tensore di diffusione costante $c = 1$ è:

$$\partial_t u = \Delta u$$

Utilizzando l'approssimazione delle derivate alle differenze finite:

$$\frac{\partial u(x,y,t)}{\partial t} = \frac{u(x,y,t+\tau) - u(x,y,t)}{\tau} + O(\tau)$$

$$\frac{\partial u(x,y,t)^2}{\partial x \partial x} = \frac{u(x+h,y,t) - 2u(x,y,t) + u(x-h,y,t)}{h^2} + O(h^2)$$

$$\frac{\partial u(x,y,t)^2}{\partial y \partial y} = \frac{u(x,y+h,t) - 2u(x,y,t) + u(x,y-h,t)}{h^2} + O(h^2)$$

dove τ è il tempo infinitesimo e h la risoluzione spaziale, otteniamo:

$$u(x,y,t+\tau) = \left(1 - \frac{4\tau}{1^2}\right) u(x,y,t) + \frac{\tau}{1^2}(u(x+1,y,t) + u(x-1,y,t) + u(x,y+1,t) + u(x,y-1,t))$$

Nella formulazione sovrastante consideriamo la risoluzione spaziale $h = 1$ poiché, avendo discretizzato le derivate, l'obiettivo è calcolare la concentrazione di ciascun pixel al tempo $t + \tau$ partendo dalla conoscenza delle concentrazioni dei pixel adiacenti al tempo t.

Abbiamo ottenuto uno schema esplicito nel quale il valore di concentrazione di un generico pixel al tempo $t + \tau$ dipende esclusivamente dal valore delle concentrazioni dei pixel adiacenti al tempo t.

Dunque per calcolare le concentrazioni di ciascun pixel attraverso il processo descritto precedentemente, è necessario risolvere un sistema di equazioni lineare con il numero di equazioni ed incognite pari al numero di pixel.

Poiché la concentrazione di un pixel dipende solo dai pixel adiacenti si otterrà una matrice dei coefficienti sparsa. Quindi è possibile risolvere il sistema attraverso metodi iterativi che sfruttino la sparsità della matrice. In particolare, nella seguente trattazione, verrà utilizzato il metodo di Jacobi (descritto nel paragrafo successivo).

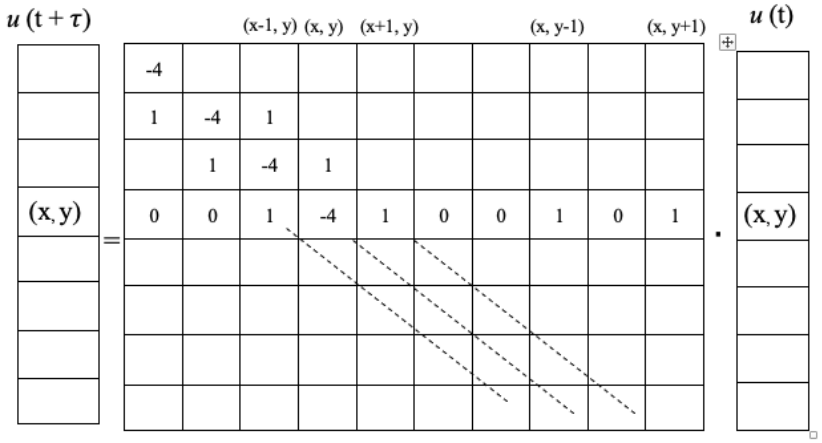


Figura 15: Rappresentazione matriciale

4.5 Metodo di Jacobi

Nella risoluzione di sistemi di equazioni lineari la scelta dell'algoritmo risolutivo più efficace richiede un'analisi delle caratteristiche del sistema. In molte applicazioni i sistemi sono di grandi dimensioni (di ordine elevato), cioè con un grande numero di equazioni e quindi di incognite, e sparsi, cioè i coefficienti di molte incognite sono uguali a zero (questo significa che ogni equazione coinvolge solo un piccolo numero di incognite).

4.5.1 Gradi di Sparsità

Dato un sistema lineare di n equazioni in n incognite, sia p il numero dei coefficienti diversi da zero. Si definisce grado di sparsità del sistema la quantità:

$$SP = \frac{n^2 - p}{n^2} \text{ con } 0 \leq SP \leq 1$$

Data una matrice A di dimensione n $\times$ m, se p è il numero degli elementi non nulli, si ha:

$$SP = \frac{n \cdot m - p}{n \cdot m} \text{ con } 0 \leq SP \leq 1$$

Se il sistema è di grandi dimensioni ed ha anche un grado di Sparsità elevato è importante utilizzare metodi che sfruttino, da un punto di vista della complessità computazionale, il fatto che la maggior parte dei coefficienti sono nulli. Per chiarire questa considerazione si supponga di voler risolvere un sistema di n equazioni in n incognite attraverso il metodo di Gauss. Sfortunatamente, la complessità del metodo di eliminazione di Gauss è, in generale, sempre la stessa, anche se il sistema presenta molti coefficienti uguali a zero. Ciò è dovuto al fatto che il metodo di eliminazione di Gauss modifica i coefficienti del sistema, durante il processo di trasformazione del sistema dato in un sistema triangolare equivalente. A causa di tali modifiche può accadere che coefficienti uguali a zero diventino non nulli. Tale fenomeno è detto **fill-in**. Il metodo di eliminazione di Gauss, almeno nella sua forma classica, non "sfrutta" il fatto che il sistema ha molti coefficienti nulli. Tale inefficienza è ancora più evidente per sistemi sparsi di grandi

dimensioni. In tal caso, una valida alternativa è costituita dai metodi iterativi. Essi consentono di sfruttare l'eventuale sparsità del sistema perché non modificano i suoi coefficienti. Si analizzano, di seguito, le caratteristiche fondamentali dei metodi iterativi e gli aspetti di base per un loro utilizzo efficiente. In particolare, tale analisi è svolta attraverso l'esame di uno tra i più noti ed antichi metodi iterativi: il metodo di Jacobi.

4.5.2 Costruzione del Metodo di Jacobi

Si consideri il seguente sistema lineare di n equazioni in n incognite:

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ \dots & + & \dots & + & \dots & + & \dots & = & \dots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n \end{cases}$$

Per risolvere tale sistema si procede nel modo seguente. Si riscrive il sistema in modo che a primo membro della i-esima equazione, $i = 1, \dots, n$, compaia solo l'incognita x_i. Questa semplice "riscrittura" è sempre possibile, eventualmente modificando l'ordine delle equazioni:

$$\begin{cases} a_{11}x_1 & = & -a_{12}x_2 & - & a_{13}x_3 & - & \dots & + & b_1 \\ a_{22}x_2 & = & -a_{21}x_1 & - & a_{23}x_3 & - & \dots & + & b_2 \\ \dots & = & \dots & - & \dots & - & \dots & + & \dots \\ a_{nn}x_n & = & -a_{n1}x_1 & - & a_{n2}x_2 & - & \dots & + & b_n \end{cases}$$

Si dividono ambo i membri della i-esima equazione del sistema scritto sopra, $i = 1, \dots, n$, per il coefficiente di x_i. Il sistema iniziale è quindi trasformato nel sistema equivalente:

$$\begin{cases} x_1 & = & \frac{1}{a_{11}}(-a_{12}x_2 & - & a_{13}x_3 & - & \dots & + & b_1) \\ x_2 & = & \frac{1}{a_{22}}(-a_{21}x_1 & - & a_{23}x_3 & - & \dots & + & b_2) \\ \dots & = & \dots & - & \dots & - & \dots & + & \dots \\ x_n & = & \frac{1}{a_{nn}}(-a_{n1}x_1 & - & a_{n2}x_2 & - & \dots & + & b_n) \end{cases}$$

Questa rappresentazione del sistema è il punto di partenza per costruire un processo iterativo: infatti, se si scelgono ad arbitrio dei valori per le incognite $x_1, x_2, \dots, x_n$, utilizzandoli nel secondo membro delle equazioni scritte sopra, si ottengono nuovi valori per $x_1, x_2, \dots, x_n$.

In generale, se $x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}$ sono i valori delle incognite $x_1, x_2, \dots, x_n$ ottenuti dopo k passi, si calcolano i nuovi valori $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_n^{(k+1)}$ mediante le equazioni:

$$\begin{cases} x_1^{(k+1)} & = & \frac{1}{a_{11}}(-a_{12}x_2^{(k)} & - & a_{13}x_3^{(k)} & - & \dots & + & b_1) \\ x_2^{(k+1)} & = & \frac{1}{a_{22}}(-a_{21}x_1^{(k)} & - & a_{23}x_3^{(k)} & - & \dots & + & b_2) \\ \dots & = & \dots & - & \dots & - & \dots & + & \dots \\ x_n^{(k+1)} & = & \frac{1}{a_{nn}}(-a_{n1}x_1^{(k)} & - & a_{n2}x_2^{(k)} & - & \dots & + & b_n) \end{cases}$$

Il procedimento descritto è, quindi, di tipo iterativo; a partire da valori iniziali arbitrari, delle incognite $(x_1^{(0)}, x_1^{(0)}, \dots, x_1^{(0)})$, si esegue una sequenza di passi, detti iterazioni, in ognuno dei quali si calcolano nuovi valori per le incognite "sostituendo" i valori calcolati al passo precedente. Si noti inoltre che ad ogni iterazione i coefficienti dell'ultimo sistema ricavato non cambiano e sono

semplicemente legati in maniera opportuna ai coefficienti del sistema iniziale. Il procedimento descritto è noto con il nome di **Metodo di Jacobi**.

Riassumendo, considerando il sistema lineare di n equazioni in n incognite come scritto all'inizio del paragrafo, questo può essere scritto nelle seguenti due forme equivalenti:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \text{ con } i = 1, \ldots, n$$

$$\text{oppure } Ax = b$$

con $A = [a_{ij}]$ matrice di dimensione $n \times n$, $b = [b_i]$ e $x = [x_i]$ vettori di dimensione n. Si assume qui e nel seguito che la matrice dei coefficienti $A \in R^{n \times n}$, del sistema definito dalle formule precedenti sia non singolare, cioè con determinante diverso da zero, e, quindi, che il sistema ammetta una ed una sola soluzione. In relazione a tale sistema, le equazioni che descrivono le iterazioni del metodo si generalizzano come segue:

$$x_i^{(k+1)} = \frac{\left(b - a_{i1}x_1^{(k)} - \ldots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k)} - \ldots - a_{i,n}x_n^{(k)}\right)}{a_{ii}} = \frac{\left(b - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right)}{a_{ii}}$$

$$\text{per } i = 1, \ldots, n \ , \ k \geq 0$$

Ovviamente, affinché tale formula siano definite è necessario che gli elementi diagonali di A, a_{ii}, $i = 1, \ldots, n$, siano diversi da zero. Se il sistema è non singolare, ciò può essere sempre ottenuto scambiando in modo opportuno le equazioni.

4.5.3 Convergenza

Si è visto che l'algoritmo relativo al metodo di Jacobi genera n successioni di valori $x_i^{(k)}$, $i = 1, \ldots, n$. Quindi, il primo problema relativo all'utilizzo di tali algoritmi è la determinazione di condizioni che assicurino che tali valori, al crescere di n, si avvicinino sempre di più alla soluzione del sistema. In altre parole, è necessario analizzare il problema della convergenza. Indicata con $x^* = [x_i^*]$ la soluzione del sistema, si vogliono determinare condizioni opportune per le quali si abbia:

$$\lim_{k \to \infty} x_i^{(k)} = x_i^* \text{ per } i = 1, \ldots, n$$

Allo scopo di illustrare le idee di base relative allo studio della convergenza del metodo di Jacobi si osservi che questo può essere espresso nella forma:

$$x_i^{(k+1)} = c_{i1}x_1^{(k)} + \ldots + c_{in}x_n^{(k)} + d = \sum_{j=1}^{n} c_{ij}x_j^{(k)} + d_i \text{ per } i = 1, \ldots, n \ , \ k \geq 0 \qquad (1)$$

O equivalentemente:

$$x^{(k+1)} = Cx^k + d, \ \ k \geq 0$$

dove $C = [c_{ij}]$ è una matrice $n \times n$ e $d = [d_i]$ è un vettore di dimensione n. È immediato verificare, dalla formula scritta nel paragrafo precedente, che per il metodo di Jacobi si ha:

$$\begin{cases} c_{ij} = 0 & \text{se } i = j \\ c_{ij} = -\frac{a_{ij}}{a_{ii}} & \text{se } i \neq j \end{cases} \tag{2}$$

$$d_i = \frac{b_i}{a_{ii}}$$

Definizione (Metodo convergente):
Si dice che un metodo del tipo scritto in precedenza è convergente se, qualunque siano i valori iniziali, le successioni $x_i^{(k)}$, $i = 1, \ldots, n$, convergono, cioè se:

$$\lim_{k\to\infty} x_i^{(k)} = \overline{x_i} \text{ per } i = 1, \ldots, n \; ; \; \forall x_i^{(0)}$$

dove $x^{(k)}$ e $\overline{x}$ sono i vettori di componenti, rispettivamente, $(x_1^{(k)}, \ldots, x_n^{(k)})$ e $(x_1, \ldots, x_n)$. Si osservi che, se il metodo converge, passando al limite in entrambi i membri della (1), si ha:

$$\overline{x_i} = \sum_{j=1}^{n} c_{ij}\overline{x_j} + d_i, \quad i = 1, \ldots, n$$

che, in forma compatta, può scriversi come:

$$\overline{x} = C\overline{x} + d$$

Se ora si considera il metodo di Jacobi e si assume che sia convergente, dalle (2) si ha:

$$\overline{x_i} = \frac{1}{a_{ii}} \left(b - \sum_{j=1, j\neq i}^{n} a_{ij}\overline{x_j} \right) = \overline{x_i} + \frac{1}{a_{ii}} \left(b - \sum_{j=1}^{n} a_{ij}\overline{x_j} \right) \text{ per } i = 1, \ldots n$$

E di conseguenza:

$$\sum_{j=1}^{n} a_{ij}\overline{x_j} = b_i \text{ per } i = 1, \ldots n$$

cioè, il limite è soluzione del sistema $Ax = b$ e, quindi, $x_i = x_i^*$, $i = 1, \ldots, n$.
Si può perciò affermare:

Proprietà
Se il metodo di Jacobi converge, allora il limite è la soluzione del sistema.

Definizione (Metodo consistente)
Un metodo del tipo (1) è consistente con il sistema di partenza se, quando converge, il limite è la soluzione del sistema suddetto. Ad ogni iterazione di un metodo iterativo del tipo (1), il vettore $x^{(k)}$ rappresenta un'approssimazione della soluzione del sistema.

4.5.4 Errore di Troncamento

Sia $x_{(k)}$ la successione generata dal metodo iterativo:

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i \text{ per } i = 1, \ldots, n \; ; \; k \geq 0$$

Si definisce errore di troncamento analitico alla k-esima iterazione del metodo, il vettore:

$$e^k = x^k - x^*$$

In base alla definizione precedente è evidente che la convergenza di un metodo del tipo (1) alla soluzione del sistema di partenza è equivalente alla convergenza della successione degli errori a zero, cioè alla condizione:

$$lim_{k \to \infty} \left| e_i^{(k)} \right| = 0 \text{ per } i = 1, \ldots, n$$

Teorema
Il metodo iterativo:

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i i = 1, \ldots, n; k \geq 0$$

è convergente se $\|C\| < 1$.

Osservazione: Il Teorema precedente stabilisce una condizione solo sufficiente per la convergenza basata sull'ordine di grandezza dei coefficienti c_{ij} in (1). Si consideri il metodo di Jacobi espresso nella forma (1), dove i coefficienti c_{ij} e d_i sono dati dalle (2). Poiché:

$$\|C\| = max_{1 \leq i \leq n} \left(\sum_{j=1}^{n} |c_{ij}| \right) = max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|}$$

Quindi se:

$$\sum_{j=1, j \neq i}^{n} |a_{ij}|/|a_{ii}| < 1 \text{ per } i = 1, \ldots, n$$

si ha $C < 1$. Un sistema di equazioni lineari in cui i coefficienti a_{ij} soddisfano le formula scritta in precedenza è detto a diagonale strettamente dominante. Si può quindi enunciare il seguente risultato:

Corollario
Se il sistema lineare è a diagonale strettamente dominante, il metodo di Jacobi è convergente.

4.5.5 Criterio di Arresto

Un aspetto fondamentale legato all'utilizzo efficace dei metodi iterativi in esame è la scelta di un opportuno criterio per decidere quando arrestare il processo iterativo. In generale, un criterio di arresto soddisfacente deve essere tale che il suo utilizzo conduca ad un risultato sufficientemente

accurato, o meglio, consenta di ottenere l'accuratezza desiderata. Appare evidente, quindi, che il criterio più naturale è richiedere che la "distanza" tra $x^{(k)}$ e la soluzione del sistema, e cioè l'errore di troncamento analitico, sia abbastanza piccola. Se si utilizza la norma del massimo per misurare la grandezza dell'errore, una possibile condizione per l'arresto del metodo iterativo è richiedere che l'errore assoluto, $\|x^{(k+1)} - x^*\|$, sia minore di una tolleranza Tol:

$$\|x^{(k+1)} - x^*\| \leq Tol$$

che rappresenta l'accuratezza richiesta.

Teorema
Dato il metodo

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i, \text{ per } i = 1, \ldots, n; k \geq 0$$

si assuma che sia consistente con il sistema lineare $Ax = b$. Se $\|C\| < 1$ si ha:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|} \|x^{(k+1)} - x^{(k)}\|, \text{ per } k \geq 0$$

Il Teorema mostra, quindi, che una stima dell'errore di troncamento analitico è data dalla differenza tra i vettori ottenuti in due iterazioni successive del metodo iterativo. Di conseguenza un possibile criterio di arresto "effettivamente utilizzabile", perché basato su quantità calcolate, è il seguente:

$$\|x^{(k+1)} - x^{(k)}\| \leq Tol$$

5 Tecniche che si basano sulla derivata prima

Il calcolo della derivata prima di un'immagine digitale si basa su approssimazioni del gradiente bidimensionale, definito per un'immagine $x(m, n)$ come il vettore

$$\nabla \mathbf{x} = \frac{\partial x}{\partial m} \mathbf{i}_m + \frac{\partial x}{\partial n} \mathbf{i}_n$$

la cui ampiezza è:

$$|\nabla \mathbf{x}| = \nabla x = \sqrt{\left(\frac{\partial x}{\partial m}\right)^2 + \left(\frac{\partial x}{\partial n}\right)^2}$$

Per semplificare il calcolo questa quantità è approssimata usando i valori assoluti:

$$\nabla x \simeq \left|\frac{\partial x}{\partial m}\right| + \left|\frac{\partial x}{\partial n}\right|$$

Questa approssimazione ancora si comporta come una derivata, cioè è nulla in aree di intensità costante e i valori risultano proporzionali al livello di variazione di intensità nelle aree non omogenee. Nella pratica è comune chiamare gradiente l'ampiezza del gradiente o una sua approssimazione. Si definisce invece direzione del vettore gradiente ∇x nel punto generico (m, n) la quantità:

$$\alpha\left(\nabla \mathbf{x}\right) = \tan^{-1}\left(\frac{\partial x/\partial n}{\partial x/\partial m}\right)$$

La direzione di un bordo nel punto (m, n) è perpendicolare alla direzione del vettore gradiente in quel punto, è infatti la direzione di massima variazione dell'intensità. Consideriamo allora un'area 3×3 che rappresenta la sezione dell'immagine che viene elaborata:

$$\begin{bmatrix} x(m-1,n-1) & x(m-1,n) & x(m-1,n+1) \\ x(m,n-1) & x(m,n) & x(m,n+1) \\ x(m+1,n-1) & x(m+1,n) & x(m+1,n+1) \end{bmatrix}$$

Esistono molti possibili filtri che approssimano la derivata se si vuole stimare il gradiente. Cominciamo con il caso più semplice.
Due semplici schemi per la derivata lungo la direzione orizzontale sono la differenza prima e la differenza centrale:

$$\frac{\partial x}{\partial m} = x\left(m,\ n\right) - x\left(m-1,\ n\right)$$
$$\frac{\partial x}{\partial m} = x\left(m+1,\ n\right) - x\left(m-1,\ n\right)$$

Entrambe queste derivate rispondono fortemente a bordi orientati lungo la direzione verticale e forniscono risposta nulla se il bordo è perfettamente orizzontale. Stesso discorso vale per derivate lungo la direzione verticale. Queste derivate possono essere espresse mediante le maschere dei filtri che le implementano nel seguente modo per la differenza prima:

$$h_1(m,n) = \begin{bmatrix} 0 & 0 \\ \mathbf{1} & -1 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} -1 & 0 \\ \mathbf{1} & 0 \end{bmatrix}$$

Mentre per la differenza centrale si ha:

$$h_1(m,n) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & \mathbf{0} & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} 0 & -1 & 0 \\ 0 & \mathbf{0} & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Il punto in grassetto indica la posizione dell'origine. In Figura 16 a destra si mostra il gradiente dell'immagine della casa raffigurata a sinistra. Si può osservare come il gradiente assume valori elevati laddove sono presenti brusche variazioni nei livelli di grigio.

Per quanto detto nel paragrafo precedente la difficoltà nell'uso di questi filtri è l'estrema sensibilità al rumore, questo può essere ridotto però inglobando l'operazione di Smoothing in ognuno dei filtri. Consideriamo per esempio la differenza centrale in una direzione e un filtro media aritmetica su 3 campioni nella direzione ortogonale. Definiamo allora le seguenti risposte impulsive:

$$h_a\left(m\right) = [1\ 1\ 1] \qquad h_b\left(n\right) = [-1\ 0\ 1]$$

Poiché h_a dipende solo da m e h_b da n, facendo il prodotto matriciale tra i due possiamo individuare un filtro bidimensionale separabile che effettua la derivata, ma realizza anche lo Smoothing:

Figura 16: Immagine Originale e Immagine Gradiente

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} [-1\ 0\ 1] = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Si ottengono in questo modo le "maschere di Prewitt":

$$h_1(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Se modifichiamo i pesi del filtro di Smoothing, otteniamo uno dei più usati rivelatori di bordi, l'operatore di Sobel:

$$h_1(m,n) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Si noti come la somma di tutti i coefficienti delle maschere sia sempre pari a zero, infatti l'operatore di derivata deve fornire un contributo nullo per aree con luminosità costante. Queste maschere possono essere modificate in modo da dare una risposta maggiore lungo la direzione diagonale per esempio per le maschere di Sobel si ha:

$$h_1(m,n) = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Il calcolo del gradiente (o meglio della sua ampiezza) permette quindi di stabilire per ogni pixel dell'immagine se sono presenti bordi valutando i massimi locali attraverso il confronto con una soglia T, tipicamente prestabilita. Se:

$$|\nabla x\,(m_0,\ n_0)| \geq T$$

allora il pixel nella posizione $(m_0,\ n_0)$ è classificato come Edge Point. Analizzando la mappa così ottenuta si può notare che l'insieme dei punti selezionati formano bordi piuttosto spessi (Figura 17 a sinistra), anziché contorni di ampiezza nulla. Per questo motivo il Thresholding

viene fatto seguire da un'operazione di Thinning che ha proprio l'obiettivo di assottigliare i bordi (Figura 17 a destra). In effetti quest'ultima operazione serve proprio a stabilire con più precisione se il pixel è effettivamente un massimo locale. La scelta della soglia è il risultato di un compromesso: una soglia elevata tende a rimuovere eventuale rumore, ma potrebbe anche eliminare dei pixel che appartengono ad un bordo per cui la mappa risulta essere caratterizzata da contorni più frammentati, di contro una soglia bassa potrebbe creare falsi Edge Point.

Figura 17: Mappa dei contorni ottenuta dopo il Thresholding e dopo un'operazione di Thinning

5.1 Filtro di Sobel

L'algoritmo di Sobel ricerca punti validi di contorno nei massimi locali della derivata prima della funzione da analizzare. Per assorbire le alterazioni della luminosità introdotte da rumore e distorsioni del sistema di acquisizione delle immagini, si stabilisce una soglia (Threshold) che può essere calcolata in modo automatico o impostata direttamente dall'operatore. Un metodo molto semplice per stabilire il valore della soglia può essere quello di valutare il rapporto tra la somma delle intensità di ogni pixel ed il numero dei pixels costituenti la matrice immagine. Un'altra peculiarità del valore di soglia è che esso può assumere un valore unico per l'intera immagine oppure adattarsi localmente. Come già scritto in precedenza, i filtri utilizzati dall'operatore di Sobel per estrarre il gradiente sono i seguenti:

$$h_1(m,n) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad h_2(m,n) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

dove h_1 estrae la componente verticale del gradiente, h_2 quella orizzontale. Come si può facilmente notare viene dato un peso maggiore agli elementi che stanno sopra ed accanto all'elemento da analizzare. Per capire il funzionamento del filtro prendiamo come esempio una matrice quadrata delle stesse dimensioni dei filtri (l'elemento x è quello da analizzare).

$$A(m,n) = \begin{bmatrix} a & b & c \\ d & \mathbf{x} & e \\ f & g & h \end{bmatrix}$$

L'azione di ogni filtro è quella di mascherare ogni elemento della matrice da analizzare con il peso relativo. Immediatamente si può dedurre che i contorni non possono essere determinati in nessuno dei punti che appartengono alla regione più esterna dell'immagine (in pratica la prima e l'ultima riga, la prima e l'ultima colonna). Il gradiente nella direzione desiderata si ottiene semplicemente sommando gli elementi della matrice filtrata. In particolare le matrici filtrate da dalle due maschere risultano:

$$A_1(m,n) = \begin{bmatrix} -1 \cdot a & -2 \cdot b & -1 \cdot c \\ 0 \cdot d & 0 \cdot \mathbf{x} & 0 \cdot e \\ 1 \cdot f & 2 \cdot g & 1 \cdot h \end{bmatrix} \qquad A_2(m,n) = \begin{bmatrix} -1 \cdot a & 0 \cdot b & 1 \cdot c \\ -2 \cdot d & 0 \cdot \mathbf{x} & 2 \cdot e \\ -1 \cdot f & 0 \cdot g & 1 \cdot h \end{bmatrix}$$

Mentre i rispettivi gradienti:

$$G_1 = -1 \cdot a - 2 \cdot b - 1 \cdot c + 1 \cdot f + 2 \cdot g + 1 \cdot h$$

$$G_2 = -1 \cdot a - 2 \cdot d - 1 \cdot f + 1 \cdot c + 2 \cdot e + 1 \cdot h$$

Dunque calcoliamo l'intensità del gradiente come:

$$G = |G_1| + |G_2|$$

e confrontiamo il valore ottenuto con la soglia. Se il gradiente risulterà essere maggiore della soglia allora il punto rispetto al quale è stato calcolato il gradiente è un punto di contorno, altrimenti no.

6 Canny Edge-detector

Il Canny Edge detector è un operatore di rilevamento dei bordi che utilizza un algoritmo a più fasi per rilevare un'ampia gamma di bordi nelle immagini. È stato sviluppato da John F. Canny nel 1986. Il Canny Edge detector è un metodo molto potente per la rilevazione dei bordi in un'immagine e cerca di migliorare il risultato dell'elaborazione sostituendo all'operazione di Thresholding i due seguenti passi:

- rendere più sottili (Thinning) i contorni direttamente sull'immagine gradiente attraverso una procedura nota come Nonmaxima Suppression;
- applicare un'operazione di Thresholding con doppia soglia (Hysteresis Thresholding).

In totale i passi da realizzare sono i seguenti:

1. l'immagine viene filtrata con un filtro gaussiano per ridurre il rumore;
2. per ogni pixel si calcola il gradiente usando per la valutazione delle derivate direzionali un filtro che è la derivata direzionale di una gaussiana;
3. per ogni pixel si stima la direzione del gradiente, e lo si classifica come Edge Point solo se lungo tale direzione il gradiente è maggiore di quello relativo ai due pixel adiacenti, altrimenti il valore del pixel è posto a zero (Nonmaxima Suppression);

4. si usa una doppia soglia allo scopo di eliminare i pixel che presentano un gradiente minore della soglia bassa (dovuto probabilmente a rumore) e conservare quelli che superano la soglia alta (classificati come Edge Point).

5. I pixel che presentano un gradiente compreso tra le due soglie vengono conservati solo se vicini ai pixel che hanno superato la soglia alta.

Data una funzione gaussiana:

$$g\left(m,n\right) = e^{\left(-\frac{m^2+n^2}{2\sigma^2}\right)}$$

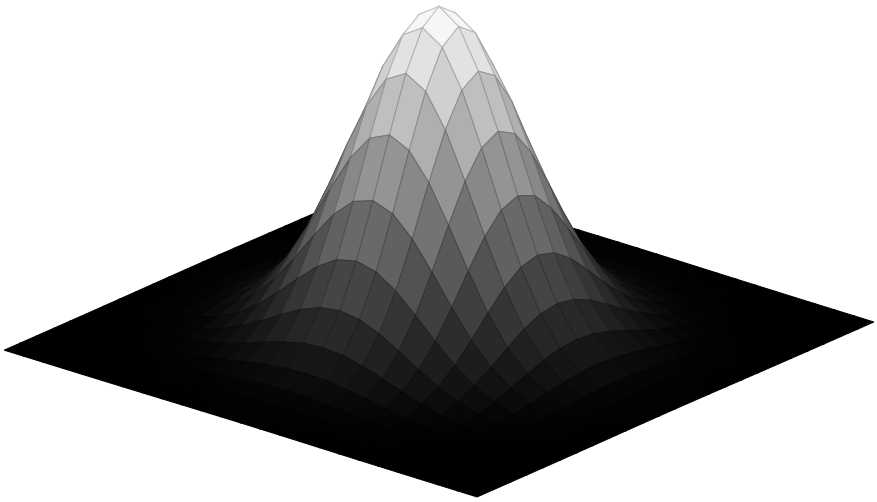


Figura 18: Rappresentazione della funzione $g(m,n)$

Le derivate lungo la direzione orizzontale e verticale di $g(m,n)$ sono:

$$\begin{cases} \frac{\partial g}{\partial m} = -\frac{m}{\sigma^2}\ exp\left(-\frac{m^2+n^2}{2\sigma^2}\right) \\ \frac{\partial g}{\partial n} = -\frac{n}{\sigma^2}\ exp\left(-\frac{m^2+n^2}{2\sigma^2}\right) \end{cases}$$

La maschera del filtro relativo alla derivata della gaussiana lungo la direzione orizzontale servirà a calcolare $\partial x/\partial m$, quello lungo la direzione verticale $\partial x/\partial n$. Quest'ultima maschera si può calcolare trasponendo la prima, data la completa simmetria esistente tra le espressioni delle due derivate. A questo punto bisogna realizzare l'algoritmo Nonmaxima Suppression, cioè annullare i valori che non sono localmente dei massimi, confrontandoli con i vicini nella direzione del gradiente. In particolare si tenga presente che intorno ad ogni pixel sono presenti 8 pixel circostanti, quindi 4 possibili direzioni, per ognuna delle quali è necessario realizzare l'algoritmo Nonmaxima Suppression. Per l'Hysteresis Thresholding è necessario fissare due soglie: T_{high} e T_{low}. Si stabilisce quindi se un pixel appartiene ad un Edge oppure no adottando la seguente strategia: se il relativo valore del gradiente

- supera la soglia T_{high}, allora è classificato come un bordo (Edge);
- è minore della soglia T_{low}, viene annullato;
- è compreso tra T_{low} e T_{high}, viene conservato solo se è vicino a pixel classificati come Edge.

Nella trattazione seguente, analizziamo in dettaglio i passaggi descritti precedentemente per realizzare correttamente l'algoritmo di Canny per il rilevamento dei bordi. Ricordiamo che tale algoritmo è composto dai seguenti cinque passaggi:

1. Riduzione del rumore;
2. Calcolo del gradiente;
3. Soppressione non massima;
4. Doppia soglia;
5. Edge Tracking per isteresi.

Dopo aver applicato questi passaggi, si otterrà il seguente risultato:

Figura 19: Immagine originale a sinistra - immagine elaborata a destra

Una prima importante considerazione da fare è che l'algoritmo si basa su immagini in scala di grigi; pertanto, un prerequisito necessario è convertire l'immagine in scala di grigi prima di eseguire i passaggi sopra indicati.

6.1 Riduzione del Rumore

Poiché la matematica coinvolta si basa principalmente sul calcolo delle derivate, in particolare sul calcolo del gradiente, i risultati del rilevamento dei bordi, come già ampiamente analizzato in precedenza, sono altamente sensibili al rumore sovrapposto all'immagine. Un modo per eliminare tale disturbo è applicare un filtro gaussiano. Per fare ciò, la tecnica di convoluzione dell'immagine viene applicata con un kernel gaussiano (3×3, 5×5, 7×7, ...).
Con il termine "kernel" si indica il nucleo di un omomorfismo, cioè l'insieme dei punti che vengono annullati dalla funzione in esame: si tratta, dunque, di uno zero-insieme. La dimensione del kernel dipende dall'effetto di Smoothing che si desidera. Fondamentalmente, più piccolo è il kernel, meno visibile è la sfocatura. Nell'esempio seguente useremo un kernel gaussiano 5×5. L'equazione per il kernel di un filtro gaussiano di dimensione $(2k+1) \times (2k+1)$ è data da:

$$K_{ij} = \frac{1}{2\pi\sigma^2}\, exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right) \quad 1 \leq i, j \leq (2k+1)$$

Equazione del kernel del filtro gaussiano

$$K = \begin{bmatrix} 0,0029 & 0,0131 & 0,0215 & 0,0131 & 0,0029 \\ 0,0131 & 0,0585 & 0,0965 & 0,0585 & 0,0131 \\ 0,0215 & 0,0965 & 0,1592 & 0,0965 & 0,0215 \\ 0,0131 & 0,0585 & 0,0965 & 0,0585 & 0,0131 \\ 0,0029 & 0,0131 & 0,0215 & 0,0131 & 0,0029 \end{bmatrix}$$

Applichiamo il filtro come definito precedentemente attuando la convoluzione tra l'immagine $x(m,n)$ ed il filtro stesso. Otteniamo così, l'immagine smoothed $y(m,n)$.

$$y(m,n) = K * x(m,n)$$

Dopo aver applicato tale filtro si ottiene il seguente risultato:

Figura 20: Immagine originale (a sinistra) - Immagine sfocata con un filtro gaussiano(a destra)

6.2 Calcolo del gradiente

Il calcolo del gradiente rileva l'intensità e la direzione dei bordi dell'immagine. I bordi corrispondono ad una variazione dell'intensità dei pixel. Per rilevarli, il modo più semplice è applicare filtri che evidenziano tale variazione di intensità in entrambe le direzioni: orizzontale (m) e verticale (n). Quando l'immagine (x) è "smoothed", vengono calcolate le derivate $\frac{\partial x}{\partial m}$ e $\frac{\partial x}{\partial n}$. Ciò può essere implementato convolvendo l'immagine x con i filtri di Sobel h_m e h_n, rispettivamente:

$$h_m = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_n = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Filtri Sobel per entrambe le direzioni (orizzontale e verticale)

Quindi, l'ampiezza del gradiente $|\nabla x|$ e la sua pendenza $\alpha(\nabla x)$ sono calcolati come segue:

$$|\nabla \mathbf{x}| = \nabla x = \sqrt{\left(\frac{\partial x}{\partial m}\right)^2 + \left(\frac{\partial x}{\partial n}\right)^2} \qquad \alpha(\nabla \mathbf{x}) = \tan^{-1}\left(\frac{\partial x/\partial n}{\partial x/\partial m}\right)$$

Intensità del gradiente e direzione del bordo

Il risultato è quasi quello previsto, ma è possibile notare che alcuni bordi sono spessi e altri sono sottili. L'algoritmo Nonmaxima Suppression sarà fondamentale per mitigare i bordi più spessi rendendo i bordi dell'immagine di spessore uniforme. Inoltre, il livello di intensità del gradiente non è uniforme, essendo compreso tra 0 e 1; i bordi finali devono avere la medesima intensità.

Figura 21: Immagine sfocata (a sinistra) - Intensità del gradiente (a destra)

6.3 Nonmaxima Suppression

Idealmente, l'immagine finale dovrebbe avere bordi sottili e di uguale spessore. Pertanto, bisogna eseguire, come già anticipato precedentemente, l'algoritmo Nonmaxima Suppression. Il principio di tale algoritmo è il seguente:

- confronta l'intensità del pixel corrente con l'intensità dei pixel posti nelle direzioni del gradiente positivo e negativo;
- se l'intensità del pixel corrente è la massima rispetto all'intensità degli altri pixel presenti nella maschera con la stessa direzione il valore verrà preservato; in caso contrario, il valore verrà soppresso.

Si consideri l'esempio seguente:

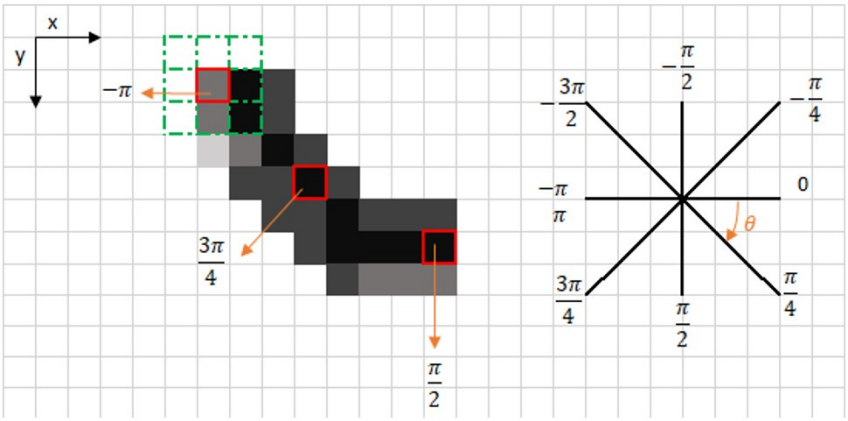


Figura 22: Esempio 1 Nonmaxima Suppression

Il riquadro rosso nell'angolo superiore sinistro presente nell'immagine sovrastante, rappresenta un pixel di intensità della matrice contenente l'intensità del gradiente. La direzione del bordo corrispondente è rappresentata dalla freccia arancione con un angolo di $-\pi$ radianti (-180°).

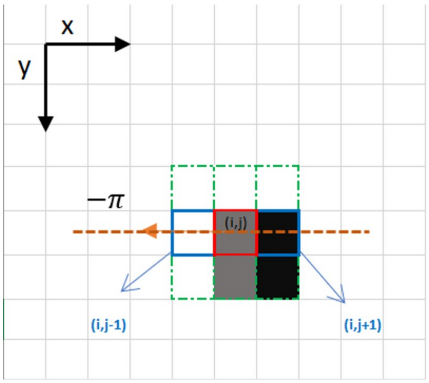


Figura 23: Focus del pixel riquadrato in rosso nell'angolo in alto a sinistra

La direzione del bordo è la linea tratteggiata arancione (orizzontale da sinistra a destra). Lo scopo dell'algoritmo è verificare se i pixel nella stessa direzione sono più o meno intensi di quelli in elaborazione. Nell'esempio, il pixel (i, j) è stato processato e i pixel nella stessa direzione sono evidenziati in blu $(i, j-1)$ e $(i, j+1)$. Se uno di questi due pixel è più intenso di quello che è stato processato, viene mantenuto solo quello più intenso. Il pixel $(i, j-1)$ sembra essere più intenso, perché è bianco (valore 1). Quindi, il valore di intensità del pixel corrente (i, j) è impostato su 0. Se non ci sono pixel nella direzione del bordo con valori più intensi, viene mantenuto il valore del pixel corrente.

Consideriamo ora l'esempio seguente:

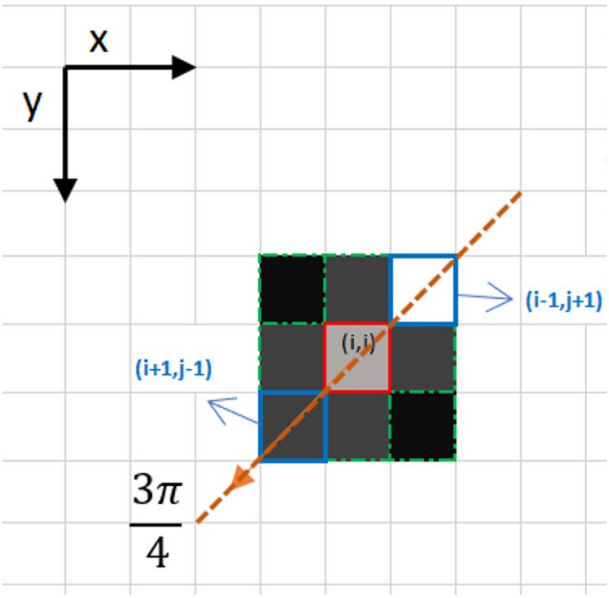


Figura 24: Esempio 2 Nonmaxima Suppression

In questo caso la direzione è la linea diagonale tratteggiata arancione. Pertanto, il pixel più intenso in questa direzione è il pixel $(i-1, j+1)$.
Riassumendo, l'obiettivo di tale algoritmo è eliminare dall'immagine modulo-gradiente i pixel che non sono massimi locali rispetto all'orientazione del gradiente. L'approccio più semplice consiste nell'analizzare l'intorno 3×3 di ogni pixel, eliminando i pixel che non rispettano la condizione di massimo locale lungo la direzione del gradiente (ortogonale all'Edge). Per verificare la condizione di massimo locale nell'intorno 3×3, si stima il modulo del gradiente nei punti adiacenti lungo la direzione del gradiente e si controlla se i pixel in tale direzione hanno un'intensità maggiore del pixel elaborato. Il pixel corrente viene conservato se la sua intensità è la massima rispetto all'intensità degli altri pixel presenti nella maschera con la stessa direzione; mentre, in caso contrario, il pixel verrà soppresso.
Il risultato, visibile nell'immagine sottostante, è la medesima immagine, ma con bordi più sottili e di uguale spessore (1 pixel). Possiamo comunque notare qualche variazione riguardo all'intensità dei bordi: alcuni pixel sembrano essere più luminosi di altri, e cercheremo di colmare questa lacuna con i due passaggi finali.

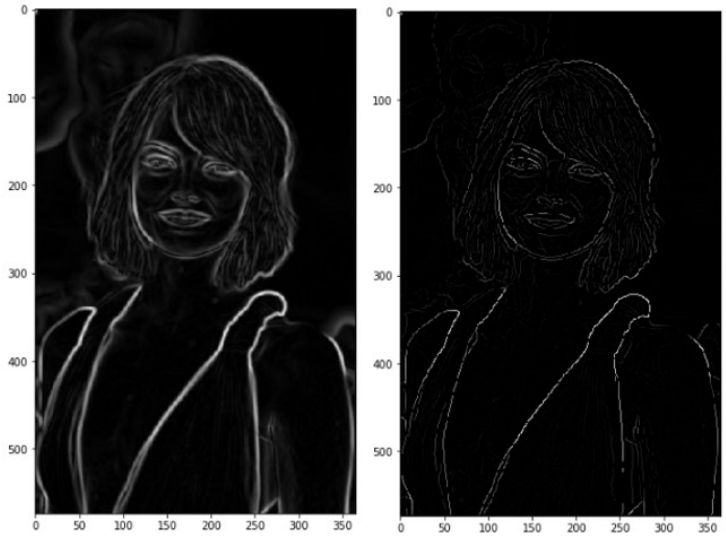


Figura 25: Risultato della Nonmaxima Suppression

6.4 Doppia soglia

Come analizzato precedentemente, per la doppia soglia è necessario fissare due soglie: T_{high} e T_{low}. Si stabilisce quindi se un pixel appartiene ad un Edge oppure no adottando la seguente strategia: se il relativo valore del gradiente:

- supera la soglia T_{high}, allora è classificato come un bordo (Edge);
- è minore della soglia T_{low}, viene annullato;
- è compreso tra T_{high} e T_{low}, viene conservato solo se è vicino a pixel classificati come Edge.

Il risultato di questo passaggio è un'immagine con solo 2 valori di intensità dei pixel:

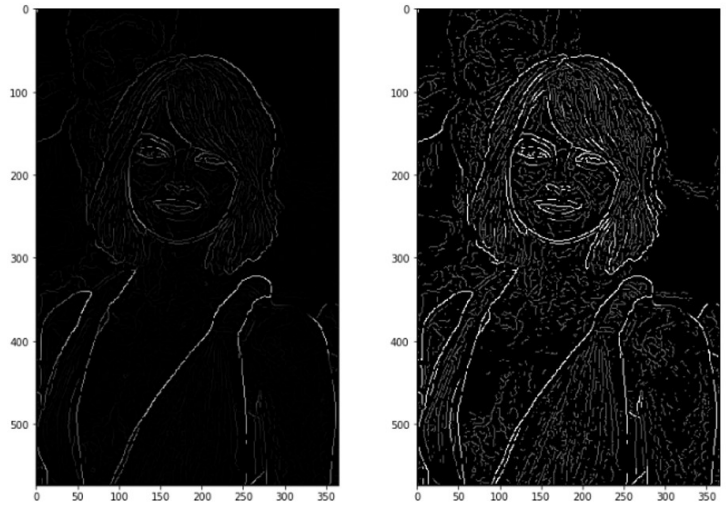


Figura 26: Immagine Nonmaxima Suppression (sinistra) - Risultato doppia soglia (destra)

6.5 Edge Tracking per isteresi

In base ai risultati della soglia, l'isteresi consiste nel trasformare i pixel con intensità compresa tra le due soglie in pixel con intensità superiore alla soglia T_{high}, se e solo se almeno uno dei pixel attorno a quello in elaborazione ha intensità superiore alla soglia T_{high}, come di seguito descritto:

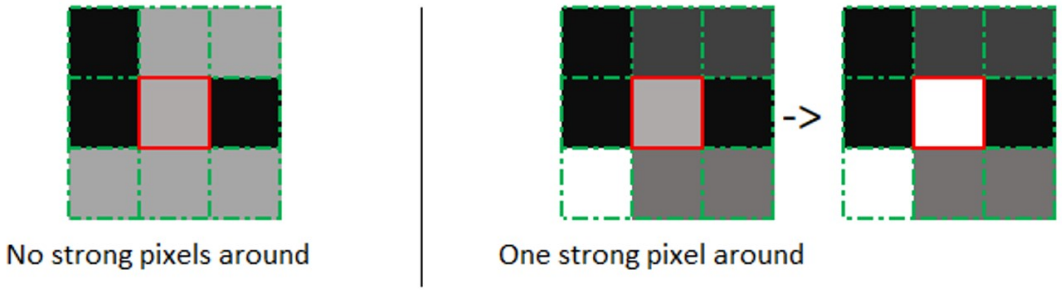


Figura 27: Esempio Edge Tracking per isteresi

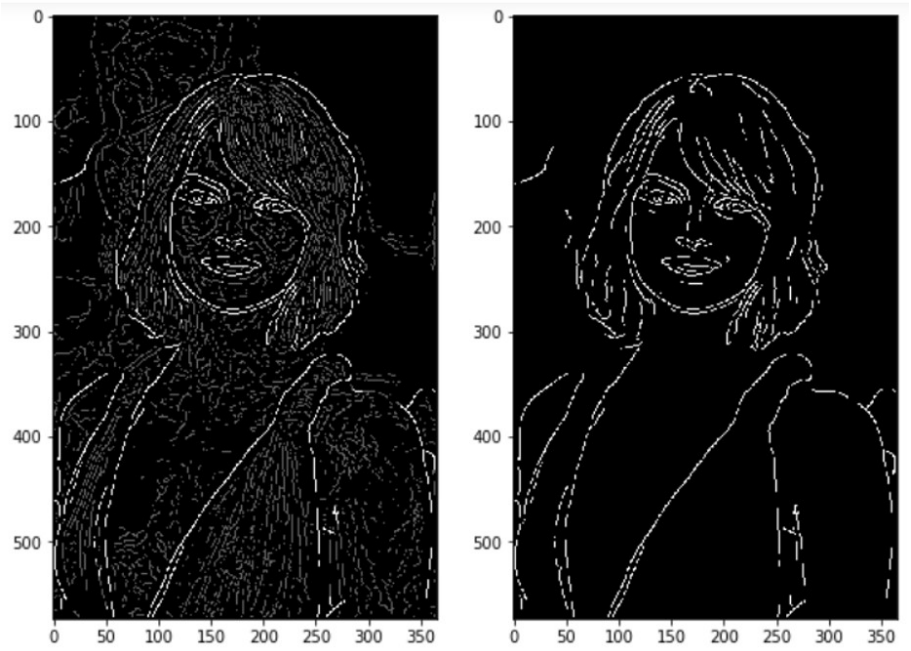


Figura 28: Risultati del processo di isteresi

7 Canny Edge-Detector per il rilevamento dei tratti significativi dei volti

Vi sono diversi approcci per il rilevamento dei tratti significativi di un volto, in particolare quello da noi considerato partirà dall'acquisizione di un'immagine contenente un volto in cui andremo ad evidenziare delle Regions of Interest (ROI). Successivamente, ad ogni regione di interesse, verrà applicato il Canny Edge-Detector al fine di individuare i bordi dei tratti significativi. Infine a tali bordi verranno associati un insieme di punti (landmarks) che verranno interpolati.

8 Individuazione delle ROI

Per individuare le regioni di interesse, viene utilizzata la funzione `vision.CascadeObject DetectorSystem`, un rilevatore di oggetti basato sull'algoritmo Viola-Jones. L'algoritmo di rilevamento di oggetti Viola-Jones è un framework di rilevamento di oggetti proposto nel 2001 da Paul Viola e Michael Jones. Sebbene possa essere progettato per rilevare una varietà di classi di oggetti, è stato motivato principalmente dal problema del rilevamento dei volti. Viola-Jones è stato progettato per volti frontali, quindi è in grado di rilevare meglio frontalmente piuttosto che volti che guardano lateralmente, verso l'alto o verso il basso. Prima di rilevare un volto, e dunque, rilevare i suoi tratti significativi, l'immagine viene convertita in scala di grigi e poi viene trovata la posizione sull'immagine colorata.

8.1 File `MostraROI.m`

Il seguente file permette di determinare le Region of Interest del volto preso in esame estrapolandole dal file *volto.jpg*. Il metodo per la determinazioni delle differenti regioni del volto (volto, occhio destro e sinistro, bocca e sopracciglia destra e sinistra) è il medesimo per ogni parte e utilizza lo stesso algoritmo precedentemente citato. Inizialmente la fotografia viene ridimensionata e quadrata tramite la funzione `imresize`. Successivamente si crea un rilevatore di oggetti tramite la funzione `vision.CascadeObjectDetector('modello')` il quale, tramite l'algoritmo di Viola-Jones, permette di rilevare la regione `('modello')` di volto da rilevare. Il parametro `'modello'` può assumere diversi valori a seconda delle regioni del volto che si vuole determinare:

- `'FrontalFaceCART'` (default): volto;
- `'LeftEye'`: occhio sinistro;
- `'RightEye'`: occhio destro;
- `'Mouth'`: bocca.

È necessario notare che la regione contenente gli occhi contiene anche le rispettive sopracciglia. Per estrapolarle è dunque necessario effettuare un ritaglio traslando il bordo inferiore dell'immagine corrispondente verso l'alto fino a raggiungere il limite inferiore del sopracciglio individuato.

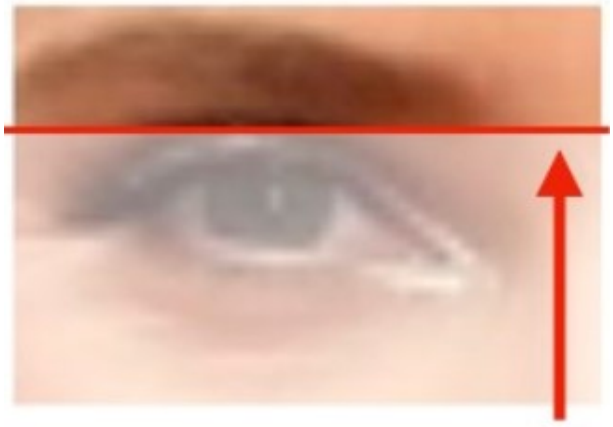


Figura 29: Estrapolazione di un sopracciglio attraverso il ritaglio

Usando la funzione `step` si utilizza il rilevatore per ricercare nella foto l'area modello da trovare. La funzione restituisce un vettore con i quattro limiti dove è contenuta la regione del modello. Tramite i valori ricavati attraverso la funzione `step` si ritaglia la fotografia e si salva il ritaglio in una nuova matrice. Viene, infine, mostrata la fotografia a video tramite la funzione `imshow`.

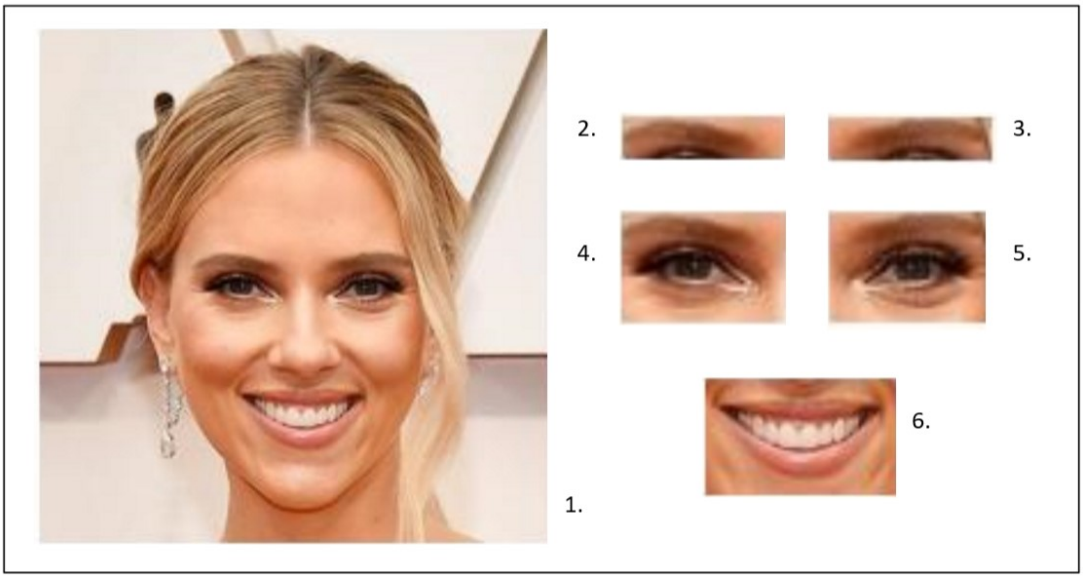


Figura 30: Rilevamento del volto(1), delle Sopracciglia(2)(3), degli Occhi(4)(5) e della Bocca(6)

8.2 File `RilevamentoROI.m`

Il seguente file effettua la stessa operazione del file `MostraROI.m`, utilizzando le medesime funzioni, senza però mostrare a video le varie zone calcolate.

9 Applicazione del metodo di Canny alle ROI

9.1 File `ProcessingSopracciglia.m`

Il passo iniziale consiste nell'equalizzare la zona delle sopracciglia ottenuta dal file `RilevamentoROI` nel seguente modo: si converte la foto in scala di grigi tramite la funzione `rgb2gray`, si normalizza l'istogramma della scala di grigi in modo tale che risulti relativamente piatto tramite `histeq`, si calcola attraverso la funzione `stretchlim` il limite superiore ed il limite inferiore dell'istogramma affinché questi possano essere utilizzati per aumentare il contrasto dell'immagine in scala di grigi e li si utilizza per migliorare il contrasto della fotografia attraverso la funzione `imadjust`.

*Fig.*1

Attraverso la funzione `imresize` si aumentano le dimensioni della fotografia di un fattore 8, sia la lunghezza (`righeSopracciglio`) che la larghezza (`colonneSopracciglio`) .

*Fig.*2

Dunque, si converte l'immagine sovrastante in B&N (bianco e nero) attraverso la funzione `imbinarize`. In particolare un generico pixel sarà nero se l'intensità di grigio è inferiore a 0.46 (su di una scala da 0 a 1), altrimenti sarà bianco. Tramite la funzione `imcomplement` si scambiano i colori: bianco in nero e viceversa.

*Fig.*3 *e Fig.*4

Si crea una matrice attraverso la funzione `bwconncomp` che etichetta con lo stesso valore gli elementi dell'immagine "connessi", ovvero simili e vicini tra loro; si noti che l'immagine è in B&N quindi 1 o 0. Quindi, si trova il massimo numero di pixel nelle regioni "connesse" tramite la funzione `regionprops` e si trovano le regioni con il massimo valore di area tramite la funzione `find`. Tramite `labelmatrix` vengono etichettati gli elementi nell'immagine "connessa"

precedentemente con `bwconncomp` con valori univoci che vengono salvati in una matrice etichetta e tramite `ismember` vengono comparati gli elementi della matrice etichetta con quelli della matrice ottenuta precedentemente con il `find`.

*Fig.*5

Si dilata l'immagine attraverso un elemento dilatante generato dalla funzione `strel`. Si dilata l'immagine nella direzione destra ($0° = 0$ nella funzione `strel`) di un fattore 10 tramite la funzione `imdilate`. Si riempiono i valori vuoti dell'immagine tramite `imfill`. Tramite `edge` viene implementato il metodo di Canny per trovare i bordi dell'immagine e quindi delle sopracciglia. Viene restituita l'immagine con 1 i bordi delle sopracciglia e 0 il resto.

*Fig.*6

9.2 File `ProcessingOcchi.m`

Il passo iniziale consiste nell'equalizzare la regione degli occhi ottenuta dal file `Rilevamento-ROI` nel seguente modo: attraverso la funzione `stretchlim` si ottiene il limite superiore ed il limite inferiore dell'immagine degli occhi affinché questi possano essere utilizzati per aumentare il contrasto dell'immagine e li si utilizza per migliorare il contrasto della fotografia attraverso la funzione `imadjust`. Tramite `imresize` si aumentano le dimensioni della fotografia di un fattore 5: sia la lunghezza (`righeOcchi`) che la larghezza (`colonneOcchi`).

*Fig.*1

L'immagine è composta dalla sovrapposizione di tre matrici corrispondenti ai colori rosso, verde e blu: la tipica configurazione Standard Red Green Blue (SRGB).
Viene creata e applicata all'immagine una struttura di trasformazione del colore ciano (C) che definisce la conversione dello spazio colore srgb tramite le funzioni `makecform` (creazione della struttura di trasformazione) e `applycform` (applicazione della struttura trasformante alla foto).

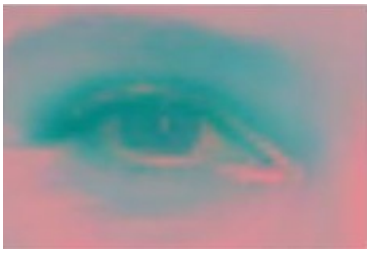


*Fig.*2

Viene calcolato un valore di soglia globale, compreso tra 0 e 1, dei livelli 2 e 3 dell'immagine tramite il metodo di Otsu con la funzione `graythresh`. Il metodo Otsu è un metodo di sogliatura automatica dell'istogramma nelle immagini digitali. L'algoritmo presume che nell'immagine da sogliare siano presenti due sole classi e quindi calcola la soglia ottima per separare queste due classi minimizzando la varianza tra esse.
Il valore di soglia viene utilizzato nella funzione `imbinarize` per convertire entrambi i livelli dell'immagine in B&N (nero se inferiore al valore di soglia, bianco altrimenti).

*Fig.*3 *e* *Fig.*4

Quindi, vengono sovrapposte le immagini.

*Fig.*5

Viene creata una nuova matrice tramite la funzione `bwlabel` che, partendo dall'immagine precedente, collega sotto medesima etichetta (1 per le regioni nere e 0 per le regioni bianche) le regioni vicine.

*Fig.*6

Tramite la funzione `imcomplement` si scambiano bianco e nero: bianco in nero e viceversa. Si crea una matrice tramite `bwconncomp` che etichetta con lo stesso valore gli elementi dell'immagine connessi tra loro, si noti che l'immagine è in B&N quindi i valori sono 1 o 0.

*Fig.*4 *e Fig.*7

Si trova, quindi, il massimo numero di pixel nelle regioni "connesse" tramite `regionprops` e si trovano le regioni con il massimo valore di area tramite `find`.
Tramite `labelmatrix` vengono etichettati gli elementi nell'immagine "collegata" precedentemente con `bwconncomp` con valori univoci che vengono salvati in una matrice etichetta e tramite `ismember` vengono comparati gli elementi della matrice etichetta con quelli della matrice ottenuta precedentemente attraverso la funzione `find`.

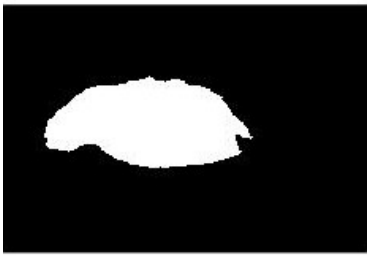


*Fig.*8

Si dilata l'immagine utilizzando un elemento dilatante creato tramite funzione `strel`. Si dilata l'immagine nella direzione destra ($0° = 0$ nella funzione `strel`) di un fattore 10 tramite la funzione `imdilate`.

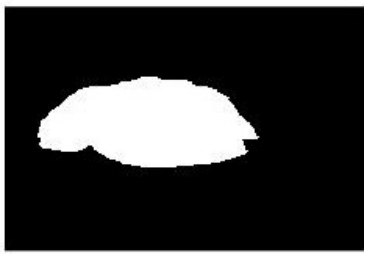


*Fig.*9

Si riempiono i valori vuoti dell'immagine tramite `imfill`. Attraverso la funzione `edge` viene implementato il metodo di Canny per trovare i bordi dell'immagine e quindi degli occhi.
Viene restituita l'immagine con 1 i bordi degli occhi e 0 il resto.

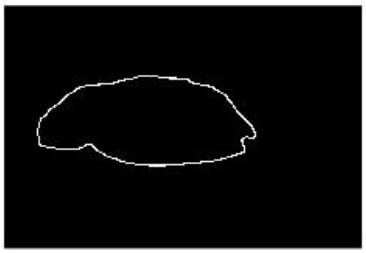


*Fig.*10

9.3 File `ProcessingBocca.m`

Inizialmente si applica un filtro Gaussiano alla regione della bocca ottenuta dal file `RilevamentoROI`. Il filtro viene generato tramite `fspecial` e lo si applica all'immagine tramite `imfilter`.

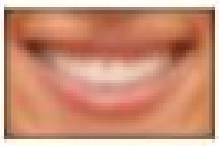


*Fig.*1

Attraverso la funzione `imresize` si aumentano le dimensioni della fotografia di un fattore 15: sia la lunghezza (`righeBocca`) che la larghezza (`colonneBocca`).

*Fig.*2

Tramite la funzione `rgb2hsv` vengono convertiti i valori di rosso, verde e blu dell'immagine RGB in valori di tonalità, saturazione e valore (HSV).

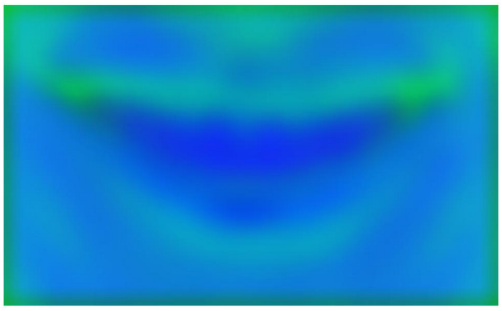
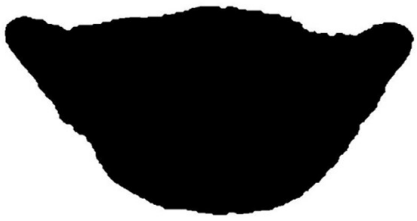


*Fig.*3

Viene erosa l'immagine utilizzando il valore di tonalità. Viene convertita l'immagine in B&N tramite `bwconncomp` e si eseguono i medesimi passaggi effettuati per le sopracciglia e per gli occhi per identificare le zone di interesse da delimitare con la funzione `edge` e il metodo di Canny.



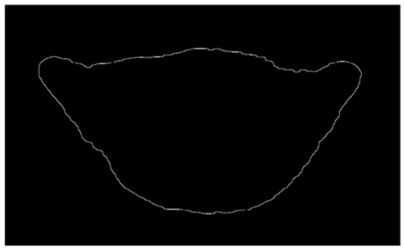
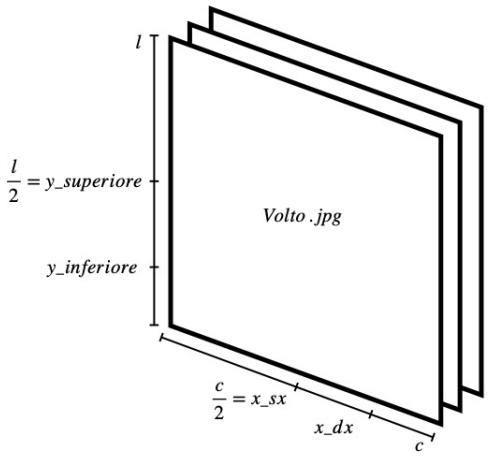


*Fig.*4 *e Fig.*5

10 Individuazione dei Landmarks

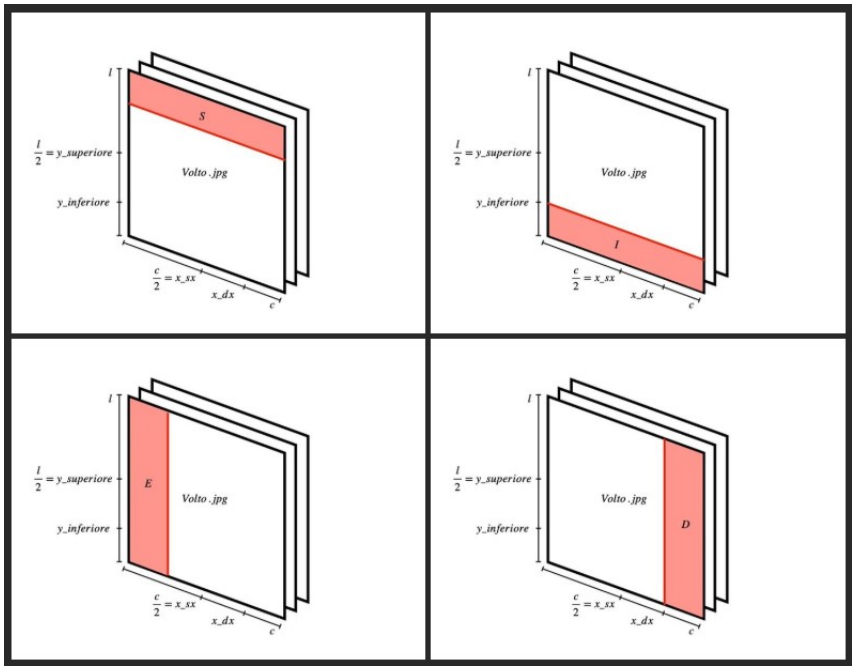
10.1 File `RilevamentoLandmarks.m`

Nel seguente file viene salvato in un array `[l, c]` la posizione dei contorni trovati per ogni zona di interesse tramite l'algoritmo di Canny.



Attraverso una serie di cicli `for` vengono esaminati tutti i punti interni all'area della fotografia processata di ogni parte del volto (la matrice canny è quella dei contorni quindi un suo generico elemento vale 1 se il punto fa parte di un contorno e 0 altrimenti) e si verifica se il punto `[i, j]` vale 1 e se si trova vicino ai bordi della zona. In tal caso vengono salvate le sue coordinate nei punti `E`, `D`, `S` e `I`. Questi rappresentano i Landmarks di sopracciglia, occhi e bocca (solo nel caso di uso di 4 Landmarks).

In caso di non esistenza di uno o più punti vengono approssimate le posizioni con le coordinate `[-1, 1]`.
Viene creato l'array `V` contenente le posizione dei punti trovati, precedentemente normalizzati al fattore di ridimensionamento usato durante il Processing delle zone (8 per le sopracciglia, 5 per gli occhi e 15 per la bocca) e viene riempito il vettore `CONT[0, 1` o `0` a seconda dell'esistenza dei punti, `0]` (il valore in `CONT(2)` è la variabile `existe` che vale 1 solo nel caso esistano tutti e 4 i punti `E`, `D`, `S` e `I`)
Vengono effettuati i medesimi calcoli per il caso di 5 Landmarks per occhi e bocca con l'aggiunta di due `if` prima del salvataggio dei punti in `V` e `CONT` che calcolano la posizione intermedia tra i punti `S` ed `I`. I seguenti punti sono quelli che appaiono al centro del rombo formato dai Landmarks di bocca e occhi. In questo caso viene riempito l'array `V` e `CONT` con la differenza che in `CONT(3)` viene inserita la posizione dei punti intermedi calcolati.



10.2 File `Distanza.m`

Il seguente file calcola la distanza Euclidea tra due punti `x` e `y` tramite Teorema di Pitagora.

11 Interpolazione polinomiale a tratti

Dopo aver applicato il metodo di Canny alle regioni di interesse per individuare i bordi dei tratti significativi del volto, l'obiettivo è quello di individuare una serie di landmarks che verranno successivamente interpolati al fine di ricostruire i tratti iniziali. Nel paragrafo successivo, si tratterà del metodo dell'interpolazione polinomiale a tratti.
Per interpolazione polinomiale a tratti si intende l'interpolazione di un set di dati con più polinomi, ciascuno dei quali definito in un sottoinsieme dell'intervallo dato. Sia $[a, b]$ un intervallo chiuso e limitato e sia la partizione dell'intervallo $[a, b]$ data da:

$$\Delta \{x_i\} \ per \ i = 0, 1, \ \ldots, k$$

su un insieme di punti, detti nodi, tali che:

$$a = x_0 < x_1 < \ \ldots < x_k = b$$

Dunque, tale metodo consiste nel posizionare con continuità parti di funzioni polinomiali. Con l'interpolazione a tratti si guadagna in flessibilità ma si perde la regolarità della funzione nei punti di raccordo, soprattutto nell'interpolazione costante a tratti, dove si creano dei gradini tra un tratto e l'altro. I polinomi derivanti dall'interpolazione polinomiale a tratti non hanno più la derivata prima continua mentre alcune volte ciò è richiesto.

Figura 31: Individuazione dei landmarks relativi alle ROI analizzate

11.1 Funzione polinomiale a tratti - Esistenza e unicità

Dato che in ogni sotto-intervallo $[x_i, x_{i+1}]$ l'interpolante richiesta esiste ed è unica, possiamo affermare che tali interpolanti relativamente ai dati $(x_i, f(x_i))$ *per* $i = 0, 1, \ldots, k$ esistono e sono uniche.
Il primo caso è quello delle interpolanti polinomiali a tratti di grado 1, cioè funzioni che in ogni intervallo $[x_i, x_{i+1}]$ *per* $i = 0, \ldots, k-1$, sono polinomi di grado $m = 1$ (e globalmente funzioni continue).

11.2 Definizione Interpolante polinomiale a tratti di grado m

Sia $x_0 = a < x_1 < ... < x_k = b$

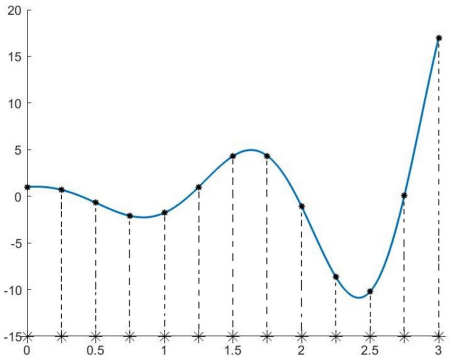


Figura 32: Partizione dei nodi di interpolazione equispaziati

Si supponga che sia $x_i = x_{i \cdot m} < x_{i(m+1)} < \ldots < x_{i \cdot (m+1)-1} < x_{k \cdot (m+1)} = x_{k+1}$, ovvero consideriamo intervalli contenenti $m+1$ punti. Nel grafico sottostante, viene mostrato il criterio attraverso il quale scegliere gli intervalli nei quali considerare le funzioni interpolanti polinomiali di grado m e vengono messi in risalto i nodi nei quali raccordare tali funzioni.

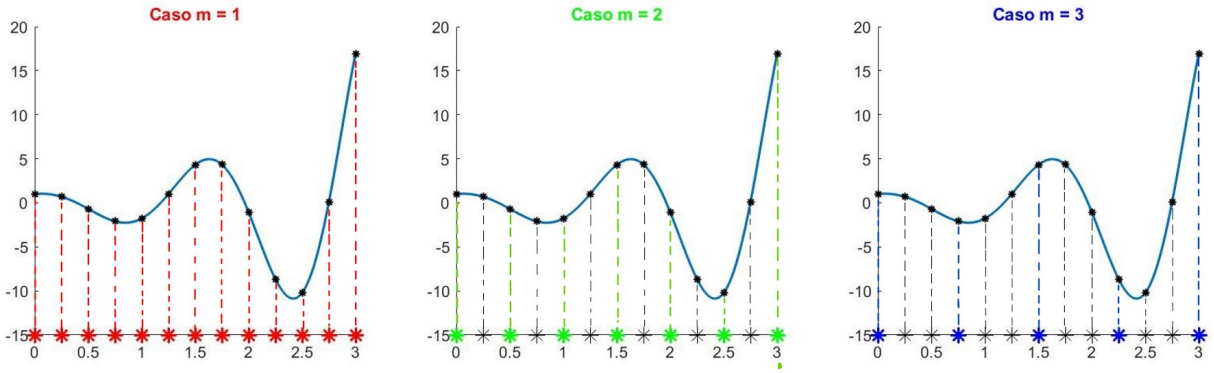


Figura 33: Suddivisione degli intervalli al variare del grado m

Si consideri la funzione p_m tale che in ogni sotto-intervallo $[x_i, x_{i+1}]$ *per* $i = 0, \ldots, k$, sia il polinomio di grado m interpolante nei punti $[x_{i \cdot m}, x_{(i+1) \cdot m}]$ i valori $[f(x_{i \cdot m}), f(x_{(i+1) \cdot m})]$. Tale p_m si chiama funzione polinomiale a tratti di grado m, interpolante le coppie $(x_i, f(x_i))$ *per* $i = 0, \ldots, k$. In altri termini p_m si ottiene posizionando con continuità parti di funzioni polinomiali ove in generale i punti di raccordo x_m, $x_{2 \cdot m}$, $x_{3 \cdot m}$, ... sono punti angolosi della funzione interpolante.

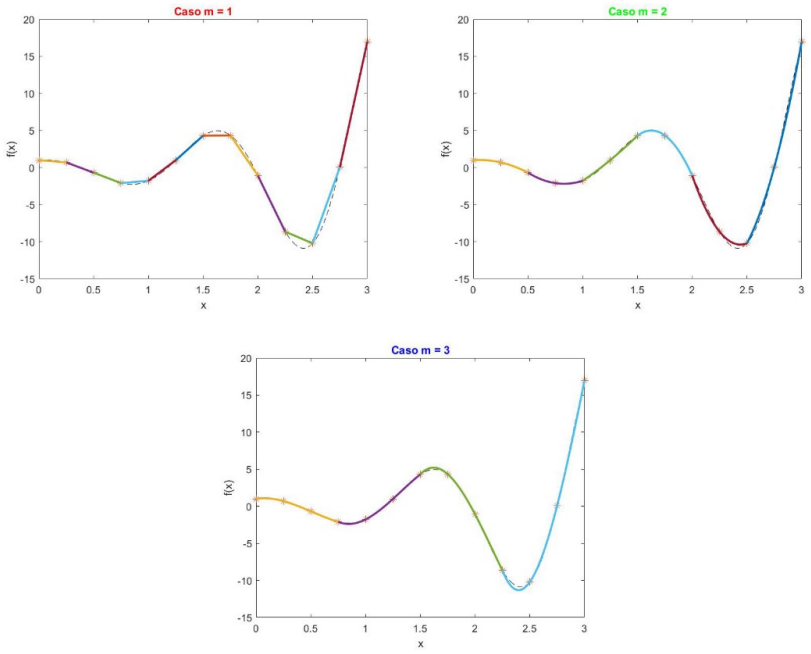


Figura 34: Esempi di interpolazione polinomiale a tratti di grado m

Nelle figure è rappresentata come, nella pratica, avviene l'interpolazione polinomiale a tratti. In particolare, nella prima immagine i polinomi interpolanti hanno grado $m = 1$, cioè le funzioni interpolanti sono delle rette e ogni retta definita nel sotto-intervallo $[x_i, x_{i+1}]$ congiunge i nodi x_i e x_{i+1}, estremi dell'intervallo in cui è definita. Nel secondo caso i polinomi interpolanti hanno grado $m = 2$, cioè le funzioni interpolanti sono delle parabole e ogni parabola definita nel

sotto-intervallo $[x_{2\cdot i}, x_{2\cdot(i+1)}]$ congiunge i nodi $x_{2\cdot i}$ e $x_{2\cdot(i+1)}$ assicurando il passaggio attraverso il nodo intermedio. Nel secondo caso i polinomi interpolanti hanno grado $m = 3$, cioè le funzioni interpolanti sono dei polinomi di 3° grado definiti nei sotto-intervalli $[x_{3\cdot i}, x_{3\cdot(i+1)}]$ congiungenti i nodi $x_{3\cdot i}$ e $x_{3\cdot(i+1)}$ assicurando il passaggio attraverso i due nodi intermedi. Bisogna notare come, nei nodi che raccordano un polinomio e il successivo la derivata non è continua, cioè tali punti sono dei punti angolosi.

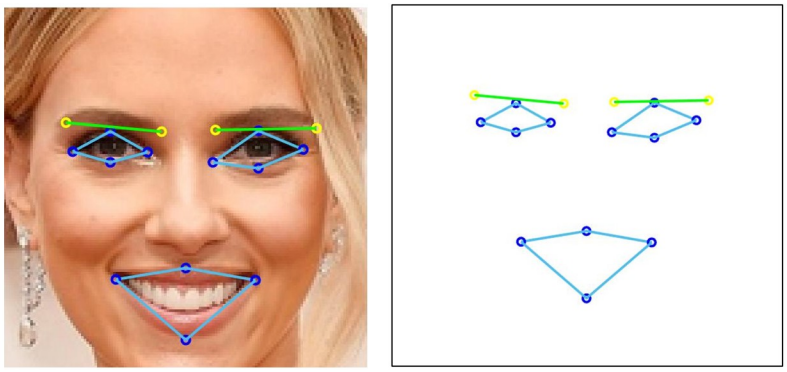
11.3 File `MostraLandmarks.m`

Nel file `MostraLandmarks.m` viene scritta una funzione che permette di plottare i landmarks e la ricostruzione dei tratti significativi del volto attraverso l'interpolazione a tratti.

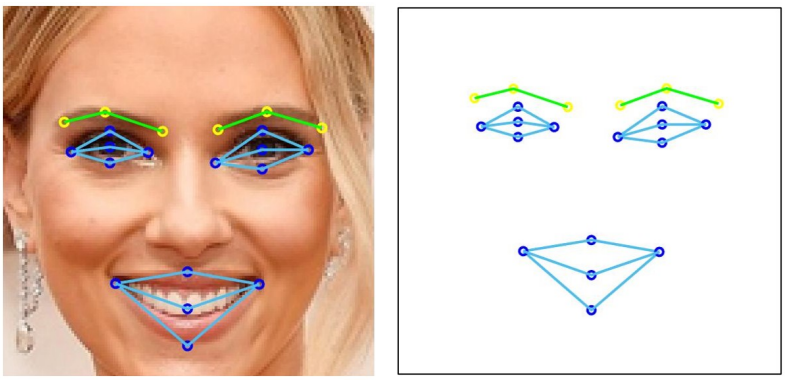
11.4 File `Ricostruzione_volto.m`

Nel file `Ricostruzione_volto.m` vengono richiamate alcune delle funzioni precedentemente descritte al fine di mostrare a video il volto analizzato e la relativa ricostruzione ottenuta. Si noti che è possibile scegliere un diverso numero di Landmarks relativi alle ROI analizzate ed è possibile scegliere se generare una ricostruzione tramite interpolazione polinomiale a tratti con polinomio interpolante di grado 1 o di grado 2. Infine è possibile scegliere se visualizzare la ricostruzione sull'immagine di partenza oppure no.

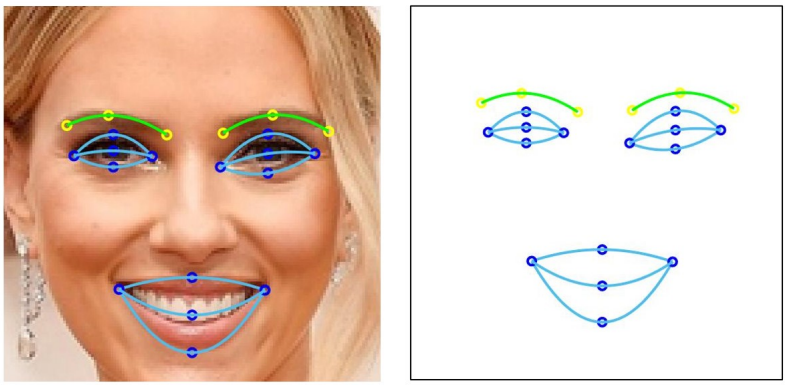
In questa prima immagine è possibile visualizzare la ricostruzione dei tratti significativi del volto precedentemente analizzati attraverso l'interpolazione lineare a tratti. Si noti come si utilizzino, come nodi di interpolazione, 2 landmarks per le sopracciglia e 4 landmarks per gli occhi e per la bocca.



Nell'immagine sottostante, invece, il numero dei nodi di interpolazione è aumentato, lasciando invariata la ricostruzione attraverso l'interpolazione lineare a tratti. In particolare si utilizzano 3 landmarks per le sopracciglia e 5 landmarks per gli occhi e per la bocca.



Infine in quest'ultima immagine si lascia invariato il numero dei nodi di interpolazione, ma la ricostruzione avviene attraverso l'interpolazione polinomiale a tratti, con polinomi interpolanti di secondo grado.



Nelle seguenti tre immagini, abbiamo provato a ricostruire i tratti significativi di un volto diverso, ma utilizzando gli stessi procedimenti descritti in precedenza.

12 Conclusioni

Dalle immagini sovrastanti si evince come la ricostruzione sia più precisa e accurata all'aumentare del numero di landmarks, cioè all'aumentare del numero dei nodi di interpolazione per ogni regione di interesse, e dunque all'aumentare del grado del polinomio interpolante. Per applicazioni dov'è richiesta una maggiore precisione, attraverso altri metodi di segmentazione dell'immagine e attraverso altri metodi di individuazione dei landmarks, si aumentano il numero di questi ultimi. Inoltre, per la ricostruzione dei tratti significativi del volto, si tende ad utilizzare un'interpolazione polinomiale a tratti a spline. Utilizzare questo metodo permette di risolvere il problema legato alla derivazione non continua.

Bibliografia

[1] R.C.Gonzalez and R.E.Woods: Digital Image Processing, 3rd Ed., Prentice Hall, 2008.

[2] A.Bovik, The essential guide to Image Processing, Academic Press, 2009.

[3] https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.html

[4] http://justin-liang.com/tutorials/canny/

[5] https://en.wikipedia.org/wiki/Canny_edge_detector

[6] Quaternoni, Salieri, Gervasio - Calcolo Scientifico - Capitolo 3: sezioni 3.4, 3.5

[7] Joachim Weickert, Anisotropic Diffusion in Image Processing, B.G. Teubner Stuttgart

[8] http://www.dma.unina.it/murli/didattica/2008_2009/cs/Cap4VolumeII.pdf