Appunti di Visione Artificiale

Riccardo Lo Iacono

Dipartimento di Matematica & Informatica
Università degli studi di Palermo
Sicilia
a.a. 2022-2023

Indice.

– 1 – Introduzione: il sistema visivo umano.

Come si vede in *Figura 1.1*, l'occhio umano ha una conformazione per lo più sferica. Si circonda da quattro membrane: *cornea* e *sclera*,che lo coprono dall'esterno, *coroide* e *retina*.

Circa la visione in se, questa è permessa da recettori luminosi posti sulla retina. Tali recettori sono distinti per struttura e funzionalità, si hanno i *bastoncelli* e *coni*.

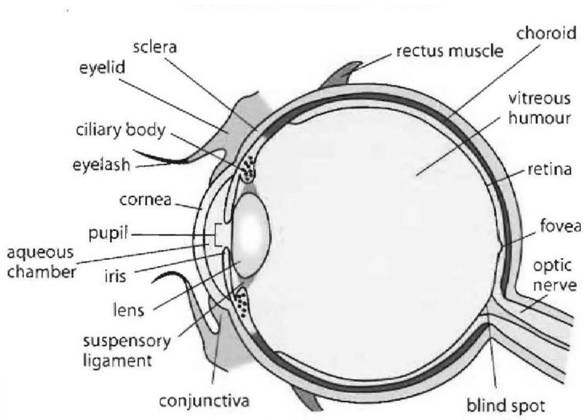


Figura 1.1: Struttura dell'occhio umano.

Analizzando le funzionalità dei due, i recettori conici sono disposti nella parte centrale dell'occhio, la *fovea*, sono molto sensibili alle variazioni di colore, e ciascun recettore è connesso ad un proprio terminale nervoso. Sono responsabili della visione *fotopica*(visione a colori). I bastoncelli, distribuiti su tutta la retina e soggetti alle variazioni luminose, connessi ad un terminale nervoso comune, hanno lo scopo di fornire un immagine generale. Sono responsabili della visione *scotopica*(visione a scala di grigi).

Osservando *Figura 1.1*, si osserva che è presente una porzione dell'occhio, quella da cui di base si estende il nervo ottico, che è priva di recettori: tale punto è detto *blind spot*, proprio perché non contribuisce alla visione. Si potrebbe pertanto pensare che la presenza di questo punto cieco, possa creare una sorta di vuoto nell'immagine. Da un punto di vista tecnico, è così. Per quel che riguarda la visione, così non è: le informazioni carpite dagli occhi giungono al cervello passando per il *chiasma*, essendo questi un "canale" comune, trasferisce in contemporanea informazioni di entrambi gli occhi, permettendo al cervello di ottenere un'immagine "pulita".

Osservazione: la visione non è globale: ossia nella realtà dei fatti ad essere messa a fuoco non è l'intera scena, quanto più una piccola porzione della stessa, quella perpendicolare alla fovea per la precisione, la nitidezza del resto dell'immagine è dovuta al cervello.

– 1.1 – Immagini digitali.

Una qualsiasi immagine digitale I, può essere vista come una funzione

$$I = \{(i, j, g) : i \in \{0, \ldots, W-1\}, j \in \{0, \ldots, H-1\}, g \in \{0, \ldots, G-1\}\}$$

dove W, H, G rappresentano rispettivamente i valori massimi di larghezza, altezza e livello di grigio[1] dell'immagine. Si deduce banalmente che la qualità dell'immagine sia dipendente dalla codifica di tali parametri. In generale si deve avere che

$$i = \min\{\lfloor W \times (x - x_{min})/(x_{max} - x_{min})\rfloor, W-1\}$$
$$j = \min\{\lfloor H \times (y - y_{min})/(y_{max} - y_{min})\rfloor, H-1\}$$
$$g = \min\{\lfloor G \times (l - l_{min})/(l_{max} - l_{min})\rfloor, G-1\}$$

[1]A meno che non sia esplicitato, saranno considerati valori di grigio nel range [0, 255].

– 2 – Teorema del campionamento e sistemi di output.

Sia assunto che l'immagine ammette frequenze massime v_x e v_y. Supponendo di dover campionare l'immagine, è così possibile determinare l'ampiezza campionante ad intervalli spaziali, dati dalle seguenti espressioni.

$$\Delta_x = \frac{1}{2v_x} \qquad \Delta_y = \frac{1}{2v_y}$$

Nel caso di pixel quadrati si impone $\Delta = \min\left\{\Delta_x, \Delta_y\right\}$.

Parlando di effettivo campionamento, si identificano principalmente tre casi, casi dai quali banalmente dipende la qualità del segnale. Questi sono

- *sotto-campionamento*: il numero di campioni del segnale da campionare, non è sufficiente a ricostruire il segnale di partenza;
- *campionamento critico*: si campiona il segnale con un numero sufficiente di campioni, permettendo di ripristinare il segnale;
- *sovra-campionamento*: il segnale è perfettamente ricostruibile, ma il numero di campioni è eccessivo.

– 2.1 – Sistema di output a scala di grigio.

Il sistema a scala di grigi che si considera è il *tubo catodico*. Questi si compone di un tubo di vetro, mantenuto a bassissima pressione, alle cui estremità sono posti due elettrodi collegati ad un generatore di corrente. Quando la differenza di potenziale tra gli elettrodi è elevata, e inoltre la pressione scende sotto le $10^{-6}\,\mathrm{atm}$, il vetro di fronte emette luminescenza. Grazie agli elettronica, con l'uso di magneti è possibile far cambiare direzione al flusso degli elettroni, secondo un percorso *raster*[2];

L'utilizzo di tale tecnologia non permetteva a volte di trasmettere a 25 fotogrammi al secondo, quantità minima di frame affinché l'immagine risulti fluida. In questi casi si procedeva con una trasmissione interlacciata: si trasmettevano cioè prima tutte le righe dispari, poi quelle pari. Motivo di tale scelta è il fatto che ad illuminarsi non è unicamente il pixel, quanto più un'areola leggermente più ampia; facendo così dunque si illuminava anche parte dei pixel delle righe pari.

– 2.2 – Sistemi di output a colori.

Davanti ciascun pixel è posta una ghiera di tre filtri: uno rosso, uno verde e uno blu. L'immagine segue sempre un percorso raster, solo che al posto di illuminare un solo pixel, procede con l'illuminare uno o più filtri.

Osservazione: i colori risultanti sono dati dalla combinazione dei tre filtri, secondo il modello RGB.

[2] L'immagine viene visualizzata a partire dal pixel più in alto a sinistra, procedendo per l'intera riga, e iniziando nuovamente dal pixel più a sinistra della riga successiva.

– 3 – Spazi-colore.

Uno *spazio-colore* è la combinazione di un modello di colore e di una appropriata funzione di mappatura di questo modello. Un modello di colore, infatti, è un modello matematico astratto che descrive un modo per rappresentare i colori come combinazioni di numeri, tipicamente come tre o quattro valori detti componenti colore. Tuttavia questo modello è una rappresentazione astratta, per questo viene perfezionato da specifiche regole adatte all'utilizzo che se ne andrà a fare, creando uno spazio dei colori.

– 3.1 – Spazio-colore RGB.

L'occhio umano possiede una visione tri-cromatica, permessa come detto in precedenza dai recettori conici. Tramite rappresentazione RGB, a ciascun pizel è associata una terna[3] di byte, potendo definire 2^{24} colori distinti. La rappresentazione di tali colori è facilmente gestibile a livello hardware.

– 3.2 – Spazio colore RGB: CCD e filtro di Bayer.

Il CCD è un dispositivo che conta quanti fotoni sono presenti in un areola, maggiore è tale numero, maggiore l'illuminazione dell'areola.

Osservazione: il CCD non è molto sensibile alle variazioni di luce, si ha quindi una soglia limite entro la quale i fotoni sono considerati.

È evidente che il CCD sinora descritto non permette che l'acquisizione di immagini in scala di grigio. Per far si che il CCD descritto permetta l'acquisizione di immagini a colore sarebbe necessaria un'areola per colore, ma ciò renderebbe difficile e costosa l'implementazione del CCD.

Per ovviare a tale problema si utilizza il *filtro di Bayer*. Sebbene ne esistano varie versioni, tutte condividono una proprietà comune: in un areola 2 x 2, due pixel sono verdi, uno rosso e uno blu. Segue che ad essere esatto è un solo colore per volta, i restanti sono ottenuti tramite media.

– 3.3 – Spazio colore HSL/HSV.

Lo spazio RGB non è l'unico spazio-colore esistente; un altro è infatti lo spazio HSV. Questi, in *Figura 3.1* rappresenta il colore, *hue*, tramite angoli: per convenzione gli zero gradi sono il rosso, i 120 il verde e i 240 il blu. Il livello di saturazione è dipendente dal *chroma*, mentre l'intensità dal *value*.

Ulteriore spazio-colore è HSL. Questi è molto simile ad HSV, infatti può essere visto come un HSV in cui tutti i colori tendenti al bianco, sono raggruppati in un unico punto.

a: Spazio-colore HSV. b: Spazio-colore HSL.

Figura 3.1: Spazi-colore HSV/HSL.

[3] Ad oggi esiste una rappresentazione che fa uso di un quarto bit, per la trasparenza il cosiddetto *alpha channel*.

– 3.4 – Spazio colore YUV.

Come tutti gli altri spazi-colore sinora descritti, anche YUV è uno spazio tridimensionale, ove le componenti Y, U, V rappresentano, rispettivamente, i livelli di luminanza e i valori di crominanza dell'immagine.

Il passaggio da YUV a RGB è effettuato tramite opportune formule trigonometriche, il viceversa tramite operazioni matriciali.

– 3.5 – Altre nozioni sugli spazi colore.

Affinché uno spazio colore sia definito tale, questi deve essere tridimensionale. Per quanto si è detto circa l'occhio umano, seguono due osservazioni.

1. La suddivisione tra canali di luminanza e crominanza di YUV, risulta sensata e utile.
2. Conseguenza del punto precedente è il fatto che, qual'ora risultasse utile comprimere l'immagine, tale compressione dovrebbe essere effettuata rispetto la crominanza. Ossia, dovendo scegliere tra il rimuovere informazioni relative la luminanza e la crominanza, è preferibile la second a.

– 4 – Operatori lineari e convoluzione.

Prima di parlare di convoluzione è necessario fare alcune puntualizzazioni. Per prima cosa si farà una distinzione tra operazione matriciale e puntuale. Si considerino le seguenti matrici

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

per il prodotto matriciale si avrebbe

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b12 + a_{22}b_{22} \end{pmatrix}$$

per quello punto-punto risulta invece

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{pmatrix}$$

Ossia nel prodotto punto-punto ad essere moltiplicati sono gli elementi i cui indici coincidono, pertanto le matrici coinvolte devono avere le medesime dimensioni

Nota: per il resto del documento saranno considerate operazioni punto-punto, se non espressamente specificato.

Ulteriore nozione è quella di operatore lineare. Si ricorda che H è un operatore lineare se

$$H[\alpha f(x,y) + \beta f(x,y)] = \alpha H[f(x,y)] + \beta H[f(x,y)] \tag{1}$$

Di interesse per l'utilizzo di MATLAB, risultano essere le seguenti operazioni lineari.

$$I^{(0)} = \{(i,j,g) : g = 0\} \implies \text{Immagine completamente nera.}$$
$$I^{(255)} = \{(i,j,g) : g = 255\} \implies \text{Immagine completamente bianca.}$$
$$kI = \{(i,j,g) : g = \min\{G-1, \lfloor k \cdot g \rfloor\}\} \implies \text{Eventuale saturazione a G}$$
$$k + I = \{(i,j,g) : g = \min\{G-1, \lfloor k + g \rfloor\}\} \implies \text{Eventuale saturazione a G}$$
$$\min\{I_1, I_2\} = \{(i,j,g) : g = \min\{g_1, g_2\}\} \implies \text{Immagine risultante è più scura}$$
$$\max\{I_1, I_2\} = \{(i,j,g) : g = \max\{g_1, g_2\}\} \implies \text{Immagine risultante è più chiara}$$
$$I_1 + I_2 = \{(i,j,g) : g = \min\{G-1, g_1 + g_2\}\} \implies \text{Eventuale saturazione a G}$$
$$I_1 \times I_2 = \{(i,j,g) : g = \lfloor (g_1 \cdot g_2)/G - 1 \rfloor\} \implies \text{Eventuale saturazione a G}$$

– 4.1 – Convoluzione.

La convoluzione è un operatore lineare, e in quanto tale soddisfa l'*Equazione 1.* Essa può essere utilizzata in vari modi, ma tutti sono accomunati da un elemento comune il *kernel.* In maniera sintetica, si pensi al kernel come una matrice i cui valori fanno da pesi alla convoluzione.

– 4.2 – Filtro di convoluzione blur.

Un filtro di convoluzione blur, o filtro di media, come suggerisce il nome, applica una sfocatura all'immagine. Concettualmente, considerata un'immagine I, si crea un kernel K con appositi valori e dimensioni, con lo scopo di creare una nuova immagine I', secondo il seguente processo.

Partendo dal pixel più a sinistra e in alto dell'immagine I, si sovrappone alla stessa K. Si procede effettuando un prodotto punto-punto tra gli elementi dell'immagine e il kernel, e se ne effettua una media. Su una nuova immagine I' si aggiunge, nelle coordinate indicate dall'elemento centrale del kernel, un pixel il cui colore è determinato dalla media precedentemente calcolata. Si ripete il processo per tutta l'immagine, spostando K secondo una logica raster.

Figura 4.1: LenaGS

Per comprendere l'effettivo funzionamento del filtro, si supponga di dover applicare a *Figura 4.1* una sfocatura. Come anticipato, la sfocatura dell'immagine è dipendete dalle dimensioni e dai valori del kernel. Dunque al fine di comprendere tale differenza siano considerate *Figura 4.2*, ottenuta tramite convoluzione con un kernel 3 x 3 i cui pesi sono tanto maggiori, tanto più vicini al centro; e *Figura 4.3* (mostrata nella sezione a seguire) ottenuta per convoluzione con un kernel 21 x 21, i cui pesi sono tutti unitari.

Partendo col considerare *Figura 4.2*, che come detto è ottenuta per convoluzione con il kernel 3 x 3 prima citato, questa risulta pressoché immutata, cosa che invece non accade in *Figura 4.3* utilizzato il kernel 21 x 21. La sfocatura è così minima da risultare impercettibile, sebbene se osservate da molto vicino risulta presente.

Sia ora considerato il seguente codice MATLAB.

Figura 4.2: Convoluzione di *Figura 4.1*, versione kernel 3 x 3.

```
% caricamento dell'immagine Lena
ker = [0 4 0; 4 8 4; 0 4 0]/24;
convLena = conv2(single(Lena), ker, 'same');
figure; imshow(uint8(convLena), [0, 255]);
```

Questi è utilizzato per ottenere *Figura 4.2.* Passando alla sua analisi, l'istruzione `ker = [...]/24;` dichiara *ker* come una matrice 3 x 3, i cui valori sono normalizzati evitando così perdita di informazioni; l'istruzione `conv2(single(Lena), ker, 'same')` effettua la convoluzione secondo quanto detto, tra l'immagine e *ker*: il parametro 'same' è utilizzato per indicare la convoluzione deve lasciare il bordo immutato, il perché sarà chiaro dopo aver letto la sezione a seguire.

Nota: il cast a single, corrispettivo del `float` in C, è necessario per via implementazione della funzione `conv2`; quello a uint8, corrispettivo dell'`inc` in C, non è strettamente necessario.

– 4.2.1 – Problema ai bordi.

Il filtro di media soffre di un grave problema, il cosiddetto *problema ai bordi.* Sebbene ottenuta con il filtro di blur, *Figura 4.2* sembra non presentare tale problema: si osservi però che il kernel utilizzato è di dimensioni minime, quindi un problema di un pixel di spessore risulta impercettibile.

Per comprendere l'effettiva presenza di tale problema, che tra l'altro non ammette soluzione concreta, si consideri *Figura 4.3.* Questa, ottenuta ottenuta eseguendo il seguente codice MATLAB, presenta evidenti differenze rispetto *Figura 4.2.*

```
% si aggiunge l'immagine trascinandola in MATLAB
ker = ones(21)/441;
convLena = conv2(double(Lena), ker, 'same');
figure; imshow(uint8(convLena), [0, 255]);
```

Figura 4.3: Convoluzione di *Figura 4.1*, versione kernel 21x21.

Tralasciando il grado di sfocatura che ovviamente risulta essere maggiore, è evidente la presenza di un bordo tendete al nero.

Si è detto che il filtro blur, come un pò tutti i filtri convolutivi, è soggetto al problema ai bordi. Ciò non è pero sorprendente: proprio per la definizione stessa di filtro di convoluzione, che come ormai noto considera il pixel identificato dall'elemento centrale del kernel, la convoluzione dei pixel al bordo non è possibile, dando così vita al problema.

Come anticipato il problema ai bordi non ammette una soluzione concreta, esistono comunque tecniche che permettono di "alleggerirlo". Considerandone alcune, queste sono le seguenti

- L'immagine viene viene estesa in ogni direzione con un bordo di pixel neri. Tale bordo deve essere sufficientemente largo da permettere di effettuare la convoluzione, a seguito della quale l'immagine è ritagliata per ottenere le dimensioni dell'immagine di partenza. È quel che si è fatto con *Figura 4.3* utilizzando il parametro 'same'.

- Si considera la convoluzione esatta: si procede con la convoluzione e si ritaglia l'immagine limitandosi a quei pixel su cui effettivamente è avvenuta la convoluzione, pertanto si riducono le dimensioni dell'immagine. Sebbene si possa considerare la scelta più corretta, si tenga a mente che all'aumentare delle dimensioni del kernel aumenta il numero di pixel di bordo rimossi.

- L'immagine è resa periodica: si ripete l'immagine in ogni direzione e si procede con la convoluzione. Ottima da un punto di vista teorico matematico, risulta pessima in quanto presenta possibili punti di discontinuità di colore, che renderebbero falsata l'immagine finale.

- Si rende l'immagine periodica secondo uno schema a mosaico: similarmente a prima l'immagine è ripetuta in tutte le direzioni, con la differenza che l'immagine viene specchiata opportunamente, procedendo successivamente con la convoluzione.

– 4.3 – Filtro mediano.

Occupandocisi di analisi di immagini digitali, capita spesso di aver a che fare con immagini soggette a rumore di varia natura. Risulta dunque necessario poter ottenere un immagine quanto più possibile priva di rumore. Passando a discutere come porre rimedio a tale problema si consideri *Figura 4.4*. Questa è un'immagine soggetta a rumore "sale e pepe", così definito perché aggiunge pixel bianchi e neri sparsi quà e là per l'immagine.

Figura 4.4: Clown soggetto a rumore.

Considerando uno dei metodi utilizzati per rimuovere il rumore, si analizza il filtro mediano. Questi è un filtro non convolutivo, che segue una logica molto semplice. Si considera una finestra W e la si fa scorrere sull'immagine, secondo logica raster, i pixel della finestra sono ordinati per tonalità di colore, e similarmente quel che si fa con un filtro convolutivo, si seleziona il pixel centrale di W e si sostituisce a questi, quello che nella sequenza dei pixel ordinati risulta essere in posizione centrale.

Osservazione: il filtro mediano non è perfetto. Questi presenta alcune problematiche, più o meno evidenti a seconda delle dimensioni della finestra. Principali problemi risultano essere quello ai bordi, e il fatto che all'aumentare delle dimensioni della finestra, l'immagine tende a sfocare, finendo col somigliare ad un filtro di media. Infine, del rumore potrebbe non essere eliminato, se la finestra è troppo piccola

Figura 4.5: Filtro mediano applicato a *Figura 4.4*.

Analizzando dunque il caso di *Figura 4.4*, applicando un'opportuna implementazione dell'algoritmo sopra citato, nel codice a seguire è fornito dall'istruzione medfilt, si ottiene quanto in *Figura 4.5*.

```
% caricamento dell'immagine con rumore
denoisedClown = medfil2(NoisyClown);
figure; imshow(denoisedClown, [0, 255]);
```

Nel caso di rumore persistente, si può provare ad adottare una o più delle seguenti tecniche.

1. Si procede con l'applicare nuovamente il filtro, aumentando opportunamente le dimensioni della finestra. Banalmente se si applicasse con le stesse dimensioni, verosimilmente l'immagine risulterebbe immutata.

2. Limitandosi unicamente ai pixel soggetti a rumore, si applica nuovamente il filtro. Utile nel caso di finestre sufficientemente grandi.

– 4.4 – Filtro di sharpening.

Altro aspetto legato all'analisi di immagini è il contrasto: si intenda questi come l'accentazione dei dettagli di un immagine. L'aumento di contrasto in un immagine digitale è effettuato tramite i filtri di sharpening. L'idea alla base è molto semplice: proprio perché ciò a cui si è interessati è esaltare i dettagli, se ad un immagine I si sottrae una sua convoluzione, eventualmente ottenuta con un filtro di media, quel che si ottiene è un'immagine D contenente i dettagli di I. Dunque se ad I si somma D, quest'ultima moltiplicata eventualmente per un qualche k, quel che si ottiene è appunto l'immagine contrastata.

Per comprende l'applicazione dei filtri di sharpening, si consideri *Figura 4.6*. Applicando ad essa il seguente codice MATLAB, che effettua lo sharpening secondo quanto descritto, quel che si ottiene è mostrato in *Figura 4.7*.

Figura 4.6: BarbaraGS.

```
% caricamento di BarbaraGS.png
ker = fspecial('average', [5, 5]);
meanB = uint8(conv2(BarbaraGS), ker, 'same');
sharp = BarbaraGS + 2.5*(BarbaraGS - meanB);
figure; imshow(sharp, [0, 255]);
```

Analizzando il codice utilizzato, la funzione `fspecial('average', [5, 5])` è equivalente all'istruzione `ones(5)/25`, cioè essa un kernel 5 x 5 i cui valori sono normalizzati. La restante parte del codice opera secondo la logica precedentemente descritta.

Figura 4.7: *Figura 4.6* sottoposta a filtro di sharpening.

Nota: nel codice utilizzato il valore di k è posto a 2.5 unicamente per permettere di apprezzare l'effettiva applicazione del filtro, in generale valori elevati sono sconsigliati.

Volendo utilizzare una versione del codice meno verbosa, quindi piu sintetica e leggibile, il codice precedentemente descritto può essere sostituito con quello a seguire.

```
% caricamento di BarbaraGS.png
sharp = imsharpen(BarbaraGS);
figure; imshow(sharp, [0, 255]);
```

Osservazione: si tenga a mente che i due codici sono del tutto equivalenti, nessuna delle due implementazioni è superiore all'altra.

– 4.5 – Filtro gradiente.

Il filtro gradiente è un filtro digitale che mette in evidenza i contorni di un'immagine, sfruttando il concetto matematico di gradiente. Per tale filtro l'immagine è considerata come una funzione a due variabili $I(x,y)$, sulla quale è appunto calcolabile il gradiente. Si ricorda che il gradiente, da un punto di vista matematico, è definito come

$$\nabla I(x,y) = \left(\frac{\partial f}{\partial x} i, \frac{\partial f}{\partial y} j\right)$$

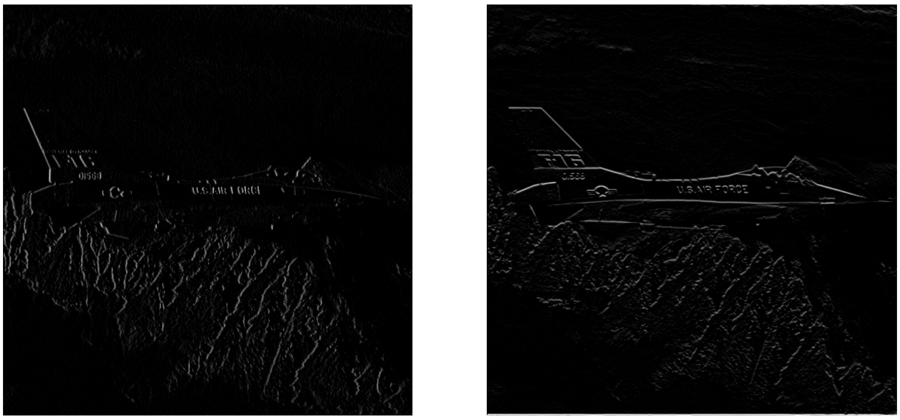
Dalla definizione continua del rapporto incrementale, si può derivare la definizione discreta delle componenti di $\nabla I(x,y)$, la quali risultano essere

$$\frac{\partial f}{\partial x} \approx I(x,y+1) - I(x,y) = \Delta_x f = \begin{bmatrix} -1 & 1 \end{bmatrix}$$
$$\frac{\partial f}{\partial y} \approx I(x+1,y) - I(x,y) = \Delta_y f = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

La definizione data risulta sensata dato che, il pixel p_1 più prossimo ad un pixel p_0 è banalmente quello ad esso adiacente.

Figura 4.8: AirplaneGS

Si consideri ora *Figura 4.8*. Procedendo dunque col calcolare le derivare in ambo le direzioni della stessa, quel che ne risulta è mostrato in *Figura 4.9*. Da questa si evince che calcolare



a: Derivata lungo x di *Figura 4.8* b: Derivata lungo x di *Figura 4.8*

Figura 4.9: Derivate *Figura 4.8*

la derivate di un'immagine lungo una direzioni, equivale a mettere in risalto i contorni lungo l'altra.

Osservazione: per una questione di visibilità, le derivate di *Figura 4.9* sono state moltiplicate per un apposita costante, mettendo così in risalto contorni che con le derivate originali sarebbero impercettibili.

Essendo il gradiente un vettore, questi ammette modulo e l'orientamento, che si ricordano essere calcolate come

$$\left\|\nabla I(x,y)\right\| = \sqrt{\Delta_x^2 f + \Delta_y^2 f} \quad \text{e} \quad \measuredangle I = \arctan\frac{\Delta_y}{\Delta_x}$$

Considerando dunque questi ultimi, è possibile calcolare la norma e l'orientamento di un'immagine. Partendo dalla norma: essendo questa la radice quadrata della somma dei quadrati delle derivate, effettuare la norma di un immagine, equivale a mettere in evidenza i contorni della stessa in ogni direzione. Quanto appena detto è mostrato in *Figura 4.10* ottenuta come norma di *Figura 4.8*.

Per quel che concerne i segmenti di codici utilizzati per ricavare le immagini di *Figura 4.9* e *4.10*, questi sono di seguito riportati nel rispettivo ordine.

```
% caricamento di AirplaneGS.png
dx = conv2(AirplaneGS, [-1, 1], 'same');
figure; imshow(dx, [0, 255]);
```

```
% caricamento di AirplaneGS.png
dy = conv2(AirplaneGS, [-1; 1], 'same');
figure; imshow(dx, [0, 255]);
```

```
% dx e dy sono le immagine sin ora calcolate
normA = sqrt(dx.^2 + dy.^2);
figure; imshow(normA, [0, 255]);
```

Figura 4.10: Contorni omnidirezionali di *Figura 4.9*

Nota: l'utilizzo di ".ˆ" sta ad indicare che il quadrato non è della matrice, ma si tratta di un quadrato punto-punto.

Passando al considerare l'orientamento, questi è ottenuto punto per punto come arcotangente del rapporto delle due derivate, volendo dunque osservare a cosa ciò equivalga, eseguendo l'opportuno codice MATLAB, a seguire, quel che si ottiene è mostrato in *Figura 4.11*.

```
% dx e dy sono le immagine sin ora calcolate
orientation = atan2(double(dx), double(dy));
figure; imshow(orientation, [-pi, pi]);
```

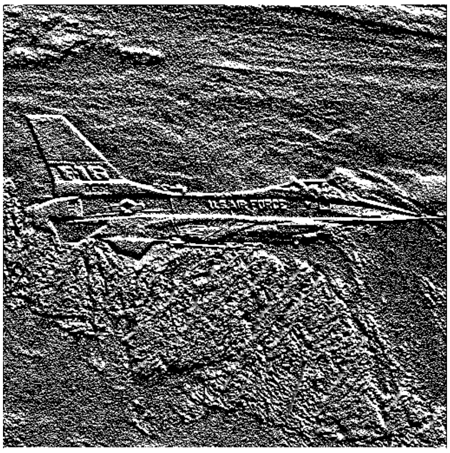


Figura 4.11: *Figura 4.9* da un punto di vista dell'orientamento dei moduli del gradiente.

– 4.6 – Filtri Prewitt e Sobel.

Considerando il filtro gradiente sin ora descritto, si osserva che i kernel utilizzati per il calcolo delle due derivate sono molto restrittivi. Nel caso dei filtri Prewitt e Sobel, anch'essi filtri che effettuano il gradiente di un immagine, il kernel è realizzato cosi da permettere una maggior flessibilità, sebbene ciò comporta quello che si può definire un "doppio bordo".

Per una questione di sinteticità: i due filtri sono analoghi del filtro gradiente precedentemente visto, differendo da questi e reciprocamente, per la matrice utilizzata come kernel. Queste risultano essere

$$\underbrace{\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{Kernel\ Prewitt} \quad \text{e} \quad \underbrace{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}}_{Kernel\ Sobel}$$

Ovviamente i kernel di cui sopra sono utilizzati per calcolare la componente Dx del rispettivo filtro, le relative trasposte quella Dy.

Si osserva inoltre che i due kernel sono a variabili separabili: cioè ottenibili come prodotto colonna-riga di opportuni vettori; nel caso dei kernel di Prewitt e Sobel si ha quanto segue.

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ Prewitt}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ Sobel}$$