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* che sono interne.

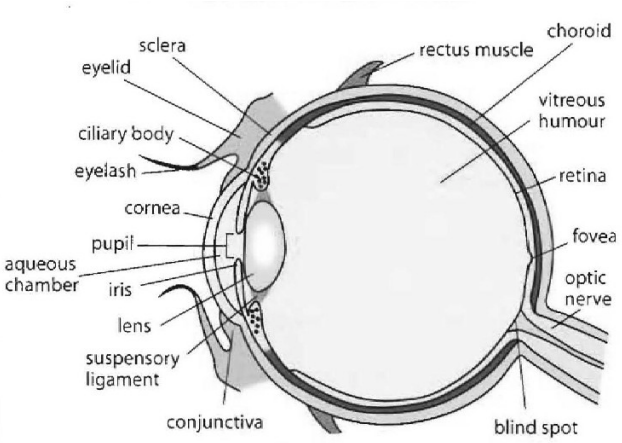

Figura 1.1: Struttura dell'occhio umano.

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*.

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

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

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

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

[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 da

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

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

Parlando di effettivo campionamento, si identificano principalmente tre casi, quali

- *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 $10^{-6}\,\text{atm}$, 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.

3.1.a: Spazio-colore HSV. 3.1.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 seconda.

– 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 matriciale risulta invece

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

Nota: per il resto del documento saranno considerate operazioni puntuali, 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 finestrella che isola parte dell'immagine.

– 4.1.1 – Filtro di convoluzione blur.

Un filtro di convoluzione blur, o filtro di media, come suggerisce il nome, sfoca l'immagine. Partendo da un immagine I, avendo un kernel K, si procede a creare una nuova immagine I', secondo quanto segue.

Si sovrappone il kernel K all'immagine I, partendo dal pixel più in alto a sinistra; si effettua un prodotto prodotto punto-punto tra il kernel e l'immagine, si sommano tali valori e lo si divide per il numero di elementi del kernel; in una nuova immagine I', nelle coordinate dell'elemento centrale del kernel, si aggiunge un pixel il cui colore è dato dalla media precedentemente calcolata. Si procede analogamente per tutti i pixel dell'immagine, spostandosi di un pixel verso destra di volta in volta e ripartendo dall'estrema sinistra della riga successiva, ogni volta che una è completata.

Si consideri *Figura 4.1*, e si supponga di dovervi applicare una sfocatura. Si consideri dunque un primo tentativo di sfocatura, utilizzando il seguente kernel

$$K = \begin{pmatrix} 0 & 4 & 0 \\ 4 & 8 & 4 \\ 0 & 4 & 0 \end{pmatrix}$$

Osservazione: Con il filtro di cui sopra la sfumatura è appena percettibile, come sarà più evidente dalle successive versioni, al cambiare del kernel, sia nelle sue dimensioni, sia nei pesi dello stesso, si ottiene una sfocatura diversa.

Figura 4.1: LenaGS

Figura 4.2: Convoluzione di *Figura 4.1*, tramite kernel K.

Eseguendo l'opportuno codice MATLAB, di seguito riportato, ciò che si ottiene è quanto in *Figura 4.2.*

```
% si aggiunge l'immagine trascinandola in MATLAB
ker = [0 4 0; 4 8 4; 0 4 0]/24;
convLena = conv2(single(Lena), ker, 'same');
figure; imshow(uint8(convLena), [0, 255]);
```

Analizzando il codice: **conv2**(single(Lena), ker, 'same') effettua l'effettiva convoluzione tra l'immagine e il kernel, dove il parametro 'same', sta ad indicare che convLena dovrà avere le stesse dimensioni dell'immagine originale; mentre **figure** ; imshow(uint8(convLena), [0, 255]) permette di visualizzare il risultato.

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

– 4.1.1.1 – Problema ai bordi.

Legato a questo filtro vi è un problema, il cosiddetto *problema ai bordi*. Si consideri ora un'alto kernel, ad esempio un kernel 11 x 11 in cui ogni elemento è posto ad 1.
Eseguendo il codice MATLAB, a seguire, ciò che ne risulta è l'immagine in *Figura 4.3*.

```
% si aggiunge l'immagine trascinandola in MATLAB
ker = ones(11)/121;
convLena = conv2(single(Lena), ker, 'same');
figure; imshow(uint8(convLena), [0, 255]);
```

Figura 4.3: Convoluzione di *Figura 4.1*, tramite kernel 11x11 unitario.

Passando all'analisi del codice: l'istruzione `ones(11)` permette di creare una matrice, quindi un kernel, 11 x 11, i cui elementi sono tutti 1.

Sebbene da *Figura 4.3* si nota appena, quel che capita è che ai bordi non è possibile applicare convoluzione, proprio perché ad essere considerato è il pixel le cui coordinate combaciano con l'elemento centrale del kernel. Inoltre, il problema peggiora tanto più grande è il kernel. Sia sin da subito chiaro che tale problema non ammette una soluzione concreta, esistono unicamente delle tecniche che permettono di "alleggerire" il problema.

Nota: si ponga l'attenzione sugli esempi di codice MATLAB, in entrambi i casi il kernel è moltiplicato per l'inverso della somma dei pesi dello stesso, effettuando pertanto una media pesata.