SCELTE IMPLEMENTATIVE

INDICE

Struttura di memoria matrice 3D

Uno dei campi presenti all'interno del tipo di dato ip_mat è *data*, ossia una variabile di tipo float*** che deve puntare ad una matrice a tre dimensioni. Uno dei nostri primi compiti è stato quello di scegliere la modalità con cui realizzare questo array multidimensionale allocato in memoria dinamica; abbiamo dunque pensato di realizzare un mix tra una matrice linearizzata e una matrice frastagliata.

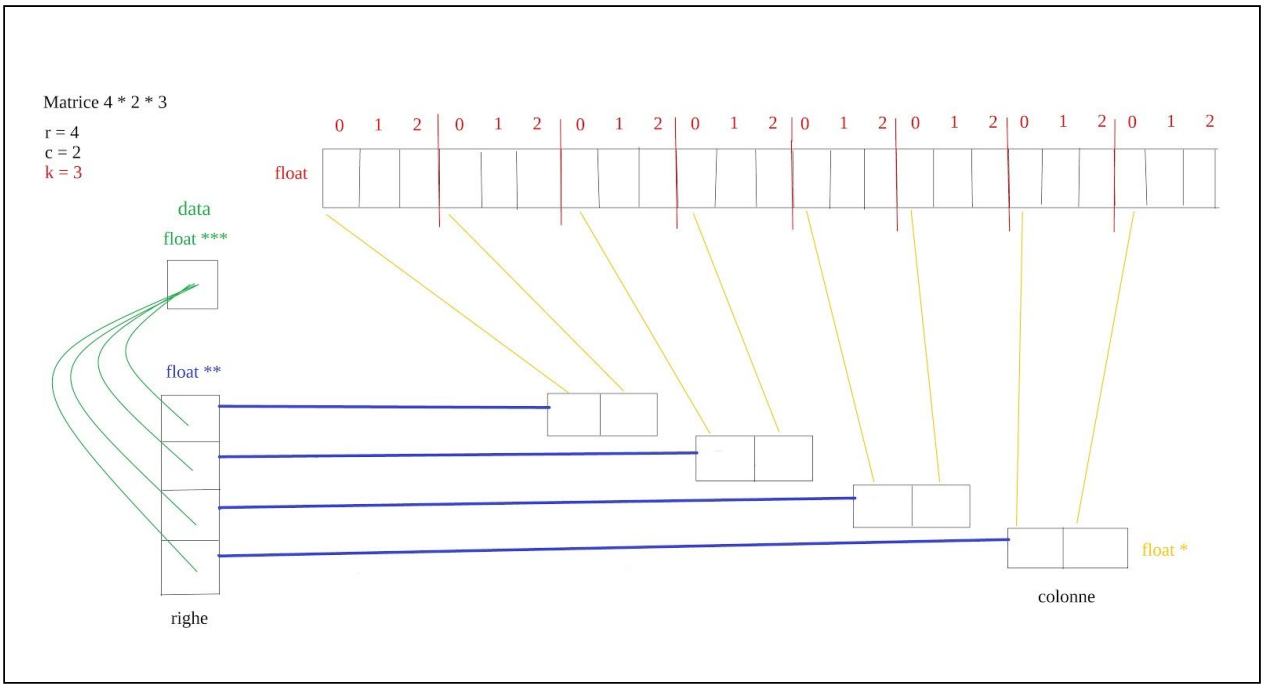
Come è possibile vedere dall'immagine sottostante la variabile data di tipo float*** punta ad un array di tipo float**, in cui il numero di elementi corrisponde al numero di righe della matrice tridimensionale che si vuole creare.

Dopodiché ciascuna cella di questo array punta ad un array di tipo float*, il cui numero di elementi corrisponde al numero di colonne della matrice tridimensionale.

Infine si presenta un array di tipo float, il cui numero di celle corrisponde al numero totale di elementi presenti all'interno della matrice 3D. Da notare che ciascun elemento appartenente ai vettori di tipo float* punta al primo valore individuato dalla combinazione riga-colonna alla quale siamo arrivati.

Es. Nel caso la matrice avesse tre canali (come nell'esempio proposto qui di seguito), ciascun elemento dell'array monodimensionale di tipo float* punterebbe ad un indice multiplo di 3 nel vettore di tipo float.

Abbiamo scelto di implementare questa tipologia di struttura per ricavare i vantaggi derivanti sia dalla matrice frastagliata sia da quella linearizzata. Infatti attraverso la frastagliata possiamo accedere subito all'elemento desiderato senza dover scorrere un'intera matrice tridimensionale partendo dal primo elemento. La linearizzata, invece, ci consente di tenere tutti gli elementi della matrice contigui in memoria dinamica.



Schema struttura di memoria matrice 3D

Funzione: gaussiana

Questa funzione dati media, varianza e un numero pseudo-casuale restituisce un valore calcolato attraverso la formula della curva gaussiana.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Abbiamo scelto di implementare questa funzione per non scrivere l'intera formula all'interno della funzione ip_mat_init_random e di conseguenza appesantire il codice e renderlo meno leggibile.

Funzione: ip_mat_corrupt

La funzione dato un valore amount e un puntatore di tipo ip_mat aggiunge del rumore, quest'ultimo viene calcolato dalla funzione get_normal_random.
Abbiamo diviso amount per due dato che la probabilità di generare numeri nel range [-amount, +amount] deve essere del 95.45% nella distribuzione.
Il valore ottenuto viene utilizzato all'interno della funzione get_normal_random per ottenere il rumore da applicare all'immagine (gauss_noise).

Funzione: convpixel

La convpixel è una funzione a supporto di ip_mat_convolve che data una matrice, un filtro e un canale restituisce il valore ottenuto dalla convoluzione delle due matrici passate come parametri.
Tale funzione viene chiamata all'interno di ip_mat_convolve passando una sottomatrice della ip_mat che si vuole modificare e delle stesse dimensioni del kernel.
Per effettuare l'intera convoluzione convpixel verrà richiamata per ogni pixel (una volta per canale).

Funzione: riempiMat

La funzione riceve in input una matrice e un vettore di float e riempie la matrice con i valori contenuti nel vettore, vi è un solo canale.
Utilizziamo riempiMat all'interno delle funzioni che fanno uso della convolve per creare i diversi filtri, evitando così di ripetere parti di codice in modo da renderlo più leggibile.

Funzione: gaussiana2v

Questa funzione dati due interi e un valore che rappresenta la deviazione standard restituisce un valore calcolato attraverso la formula della curva gaussiana.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Abbiamo scelto di implementare questa funzione per non scrivere l'intera formula all'interno della funzione create_gaussian_filter e di conseguenza appesantire il codice e renderlo meno leggibile.