Traitement d'image :

Opérations élémentaires sur les images

A.Mazza
MPSI 2-3
CPGE Oujda

21 Avril 2025

Convolution

La plupart des filtres de traitement d'images utilisent une convolution par un masque pour réaliser une modification. Ces masques sont des matrices de petite taille, en général 3 × 3 (taille que nous allons considérer par la suite) ou 5 × 5 :

$$C = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

Le filtre associé à ce masque transforme la matrice $M = (m_{ij})$, associée à une certaine image, en la matrice $M \otimes C = (m^*_{ij})$ par une opération appelée **produit de convolution**, définie par la relation :

$$\forall i, j,$$

$$m^*_{ij} = c_{11} m_{i-1,j-1} + c_{12} m_{i-1,j} + c_{13} m_{i-1,j+1} + c_{21} m_{i,j-1} + c_{22} m_{i,j} + c_{23} m_{i,j+1} + c_{31} m_{i+1,j-1} + c_{32} m_{i+1,j} + c_{33} m_{i+1,j+1}$$

Convolution

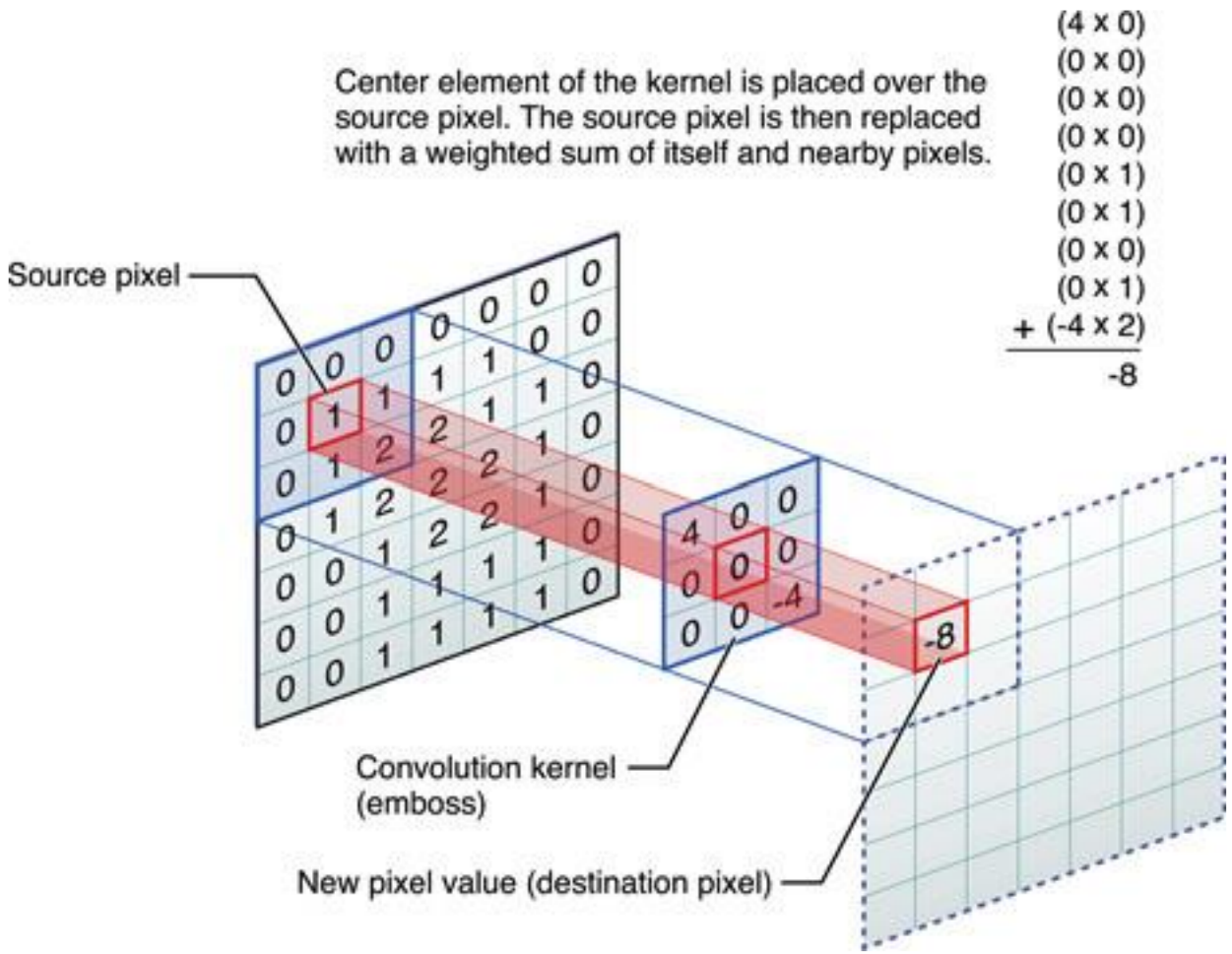

Figure: Convolution

Convolution

Dans le cas d'une image en couleur, on peut choisir d'appliquer le même masque à chacune des trois composantes RGB de l'image, ou leur appliquer des masques différenciés, en fonction de l'effet désiré.
Dans la suite de ce sujet, et dans un but simplificateur, nous appliquerons le **même masque** à chacune des trois composantes de couleur.
Remarque. Lorsque le pixel initial est situé sur un bord, une partie de la convolution porte en dehors des limites de l'image. Là encore, nous conviendrons, pour simplifier, de **laisser inchangés ces pixels**.

Exercice

Exercice : Rédiger une fonction `convolution` qui prend en arguments deux matrices M (associée à une image) et C, et retourne la matrice $M \otimes C$.
Les éléments de la matrice C seront, a priori, de type `float` tandis que ceux des matrices M et $M \otimes C$ seront de type `np.uint8`.
Il conviendra donc, lorsque $m^*_{ij} < 0$ ou $m^*_{ij} > 255$, de remplacer les valeurs calculées par respectivement 0 et 255.
Ne pas oublier non plus que les pixels d'une image en couleur possèdent trois composantes (R, G, B) et que chacune de ces composantes doit être traitée.

Exemples classiques de matrices de convolution

- **Flou (blur)** Pour flouter une image (moyenne locale) :

$$C = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- **Détection de contours (Sobel)** Pour détecter les contours horizontaux :

$$C_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Pour détecter les contours verticaux :

$$C_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Exemples classiques de matrices de convolution

- **Renforcement des contours (sharpness)** Pour accentuer les détails de l'image :

$$C = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

- **Détection de bords (Laplacien)** Utilisé pour détecter les changements rapides d'intensité :

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

ou parfois :

$$C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Exercice : Réduction d'une image par moyennage local

Soit une image T représentée sous forme d'une matrice tridimensionnelle de taille $(n_l, n_c, 3)$, où n_l est le nombre de lignes, n_c est le nombre de colonnes, et chaque élément de la matrice contient les valeurs des trois canaux de couleur (R, G, B) pour chaque pixel.
On souhaite réduire cette image en la subdivisant en blocs de taille $r \times r$, et en affectant à chaque bloc une nouvelle valeur de couleur, qui est la moyenne des couleurs de tous les pixels contenus dans le bloc.
L'objectif est de créer une fonction `reduction` qui réduit une image T d'un facteur r (facteur entier). Cette fonction calcule la moyenne des pixels dans chaque bloc $r \times r$ et affecte cette moyenne à un pixel unique dans l'image résultante.
Question : Implémentez une fonction `reduction` en Python, qui prend en argument une image T et un facteur de réduction r, et retourne une image réduite. La fonction doit suivre les étapes suivantes :

- Diviser l'image T en blocs de taille $r \times r$.
- Calculer la moyenne des couleurs pour chaque bloc (c'est-à-dire la moyenne des composantes R, G et B de tous les pixels du bloc).
- Créer une nouvelle image où chaque pixel correspond à la moyenne des couleurs du bloc $r \times r$.

Exercice : Agrandissement d'une image par duplication locale

Soit une image T représentée sous forme d'une matrice tridimensionnelle de taille $(n_l, n_c, 3)$, où n_l est le nombre de lignes, n_c est le nombre de colonnes, et chaque élément de la matrice contient les valeurs des trois canaux de couleur (R, G, B) pour chaque pixel.
On souhaite agrandir cette image en la transformant en une nouvelle image où chaque pixel de T est remplacé par un carré de $r \times r$ pixels de même couleur.
L'objectif est de créer une fonction `agrandissement` qui prend une image T et un facteur r (facteur entier), et retourne l'image agrandie.
Question : Implémentez une fonction `agrandissement` en Python, qui prend en argument une image T et un facteur d'agrandissement r, et retourne une image agrandie en répétant chaque pixel de T en un bloc de $r \times r$ pixels identiques.
Explications :

- **Dimensions de l'image agrandie** : La nouvelle image aura $n_l \times r$ lignes et $n_c \times r$ colonnes.
- **Duplication des pixels** : Chaque pixel original est copié en un bloc de $r \times r$ pixels de même valeur.
- **Gestion des canaux de couleur** : L'agrandissement est réalisé pour chacun des trois canaux (R, G, B) séparément.