TD/TP 4 : Convolution

Version du 20 août 2015

1 Introduction

Il existe en traitement d'image une opération très importante : le filtrage par convolution. De quoi s'agit-il ? Une convolution est l'opérateur noté $\star$ et définie de la façon suivante :

$$f \star g(x) = \int f(x-y)g(y)dy \tag{1}$$

où f et g sont deux fonctions réelles intégrables. Pour les fonctions de $\mathbb{R}^2$, la définition devient :

$$f \star g(x', y') = \int\int f(x'-x, y'-y)g(x,y)dxdy \tag{2}$$

Dans l'espace discret où vivent les images, la formule s'écrit :

$$f \star g(k,l) = \sum_i \sum_j f(k-i, l-j)g(i,j) \tag{3}$$

On supposera que nos fonctions ont un support (i.e. un domaine de définition) borné. Ainsi, les indices i et j varient dans un intervalle fini.

En particulier, on suppose f définie sur $[1, N] \times [1, P]$ et g définie sur $[-\frac{n}{2}, \frac{n}{2}] \times [-\frac{p}{2}, \frac{p}{2}]$, la formule (3) devient :

$$f \star g(k,l) = \sum_{\substack{1 \leq k-i \leq N \\ -\frac{n}{2} \leq i \leq \frac{n}{2}}} \sum_{\substack{1 \leq l-j \leq P \\ -\frac{p}{2} \leq j \leq \frac{p}{2}}} f(k-i, l-j)g(i,j) \tag{4}$$

On remarque immédiatement que cette définition est plus compliquée que la précédente. En effet, la convolution revient, entre autre, à translater une des deux fonctions. Sur un support infini, la translation n'a pas d'effet sur le support. En revanche, avec un support fini, le domaine est également translaté. La conséquence est donc que le support de la convoluée est différent des supports des fonctions convoluées. Dans l'exemple de l'équation 4, le support de $f \star g$ est $[1 - \frac{n}{2}, N + \frac{n}{2}] \times [1 - \frac{p}{2}, P + \frac{p}{2}]$. La figure 1 illustre cela : le calcul du pixel en bas à droite est possible ; sa valeur est a priori non nulle parce qu'elle dépend de la valeur d'un pixel de I (celui en gris), les autres étant nuls.

2 Applications aux images

En pratique, nous l'avons déjà dit, la convolution d'image est une opération très importante. Soit I une image de taille $N \times P$, $I(x, y)$ est donc la valeur de luminescence du pixel (x, y). Cette image peut être *convoluée* avec une autre image k que l'on appelle *noyau (ou kernel en anglais) de convolution*. Le noyau k est généralement de petite taille (typiquement 3×3, 5×5 ou 7×7) et représente souvent un opérateur différentiel (mais pas nécessairement), il est représenté centré en 0.

Par exemple, si l'on souhaite calculer la dérivée en x de I ($\frac{\partial I}{\partial x}$), on peut approcher par différence finie (développements de Taylor à l'ordre 1) l'opérateur :

$$\frac{\partial I(x,y)}{\partial x} \sim \frac{1}{2}(I(x+1, y) - I(x-1, y)) \tag{5}$$

ce qui revient à calculer $I \star k$ avec :

$$k = \begin{pmatrix} 0 & 0 & 0 \\ -1/2 & 0 & 1/2 \\ 0 & 0 & 0 \end{pmatrix} \tag{6}$$

Avec d'autres opérateurs, on peut lisser une image, extraire des contours, *etc*. On peut voir la convolution comme une opération de moyennage pondérée sur un voisinage fixé par le noyau de convolution (voir figure 1).

3 Algorithme séquentiel

Avec la définition donnée de la convolution discrète (équation (4)), on ne peut pas calculer la convolution sur les pixels au bord de l'image car elle nécessite l'accès à des pixels non définis, pourtant le calcul sur les bords est possible (voir la figure 1, en bas à gauche et en bas à droite).

Pour résoudre ce problème, il y a deux méthodes :

— soit on agrandit artificiellement la taille de l'image, les nouveaux pixels ont alors une valeur nulle,

— soit on exclut du calcul de convolution les pixels sur les bords de l'image.

Notre algorithme choisit la deuxième méthode car notre propos n'est pas d'implémenter rigoureusement la convolution mais de la paralléliser.

Pour obtenir de longs temps de calcul et illustrer les envois de données de MPI, l'opération de convolution est itérée un grand nombre de fois. On veut donc implémenter l'opération :

$$I \star^n k = I \star \underbrace{k \star k \cdots \star k}_{n \text{ fois}} \tag{7}$$

Comme nombre d'applications numériques (comme la résolution d'équations aux dérivées partielles) requièrent la répétition d'un grand nombre de convolutions, ce choix est pertinent.

Finalement, l'algorithme est simple :

1. Lecture de la ligne commande (protocole Argv).
2. Chargement de l'image (fonction lire_rasterfile).
3. Répéter `nbiter` fois la convolution de l'image (« pointée » par le champ `r.data` de la variable `r` de type `Raster`) par un des 5 filtres :
   (a) allocation mémoire d'un tampon image intermédiaire,
   (b) pour chaque pixel (i, j) de l'image (sauf les bords) faire :
      — le pixel (i, j) de l'image tampon reçoit une combinaision linéaire (c'est l'opération de convolution) du pixel (i, j) et de ses 8 plus proches voisins dans l'image (`r.data`).
   (c) recopie du tampon intermédiaire dans l'image (`r.data`) et libération mémoire du tampon image intermédiaire.
4. Sauvegarde de l'image (fonction sauve_rasterfile).

Les 5 différentes combinaisons linéaires représente les cinq filtres possibles (voir l'instruction switch dans le listing) : ces opérations sont dans l'ordre : un filtre moyenneur, un autre filtre moyenneur (avec plus de poids au centre), un détecteur de contour (c'est l'opérateur Laplacien discrétisé par différences finies : un tel opérateur donne une forte réponse sur les contours de type "pic", voir figure 2), un autre détecteur de contour qui détecte les contours de type "marche" (voir figure 2). et finalement un filtre médian (non linéaire : on trie les pixels selon leur luminosité et on retient la valeur médiane).

4 Questions

1. Dans la fonction `convolution()`, pourquoi doit-on préparer un tampon intermédiaire au lieu de faire le calcul directement sur l'image ?
2. Quelles sont les séquences parallélisables de l'algorithme ?
3. Sachant que la taille du noyau k de convolution est de 3×3 pixels, quelle est la complexité théorique du calcul d'un pixel de $I \star k$? Quel type d'équilibrage de charge doit-on prévoir entre les processeurs ?
4. Quel découpage (répartition des données entre processeur) est naturel dans ce contexte ?
5. Quel problème (aux bords des blocs d'image) survient lors de l'itération de l'opération de convolution ?
6. Implémenter un algorithme parallèle avec des envois bloquants de messages.

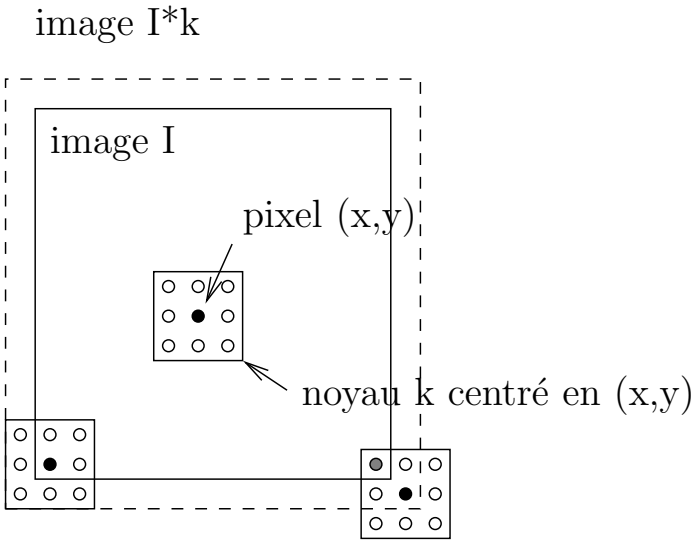

FIGURE 1 – Convolution de I par k : $I \star k(x, y)$ est la somme pondérée (par k) des valeurs de I en (x,y) et de leurs voisins. On illustre aussi le fait que le domaine de $I \star k$ (en pointillé) est le domaine de I augmenté de celui de k. Au-dela, les valeurs sont nulles.

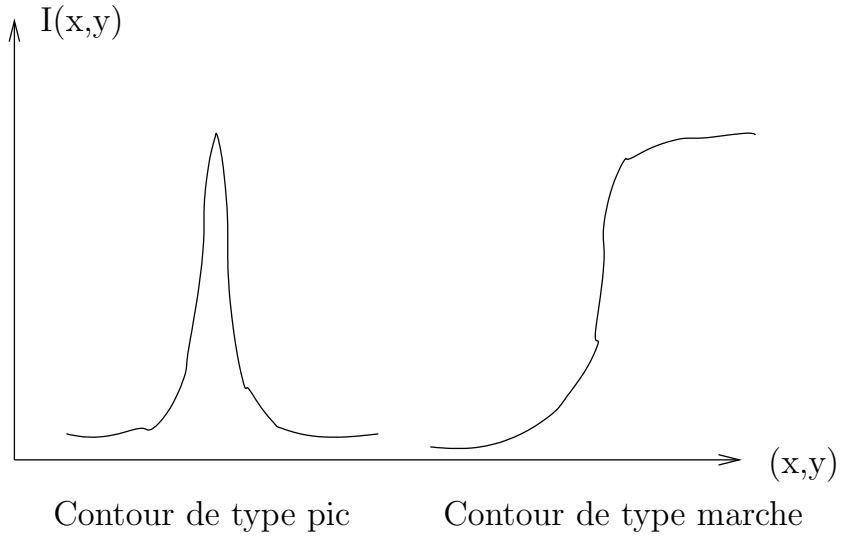

FIGURE 2 – Exemple de contours.

7. Question subsidiaire : implémenter l'algorithme avec cette fois des primitives non-bloquantes et de façon à ce que les temps de calcul recouvrent les temps de communication. Analyser les performances obtenues.

