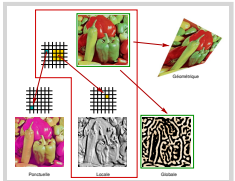
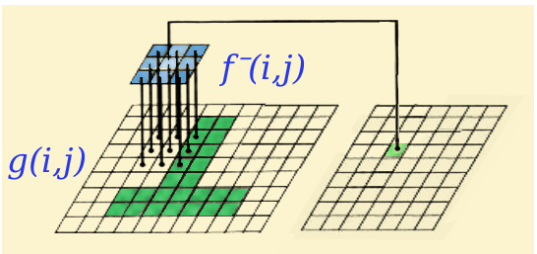


transformation locale

Partie B — thème 4

Convolutions discrètes 2D



1. Rappels

1.1. Définition

$$(f * g)(i,j) = \sum_{i'=-D_i}^{D_i} \sum_{j'=-D_j}^{D_j} f(i-i', j-j')\, g(i',j')$$

- pour chaque pixel P : $\begin{cases} \text{translation du symétrique de } f \text{ sur } P \\ \text{somme des produits terme à terme} \end{cases}$
- en pratique, le noyau f est une matrice rectangulaire de petite taille

$$(2D_i + 1) \times (2D_j + 1)$$

exemple pour $D_i = 2$ et $D_j = 3$:

- si le noyau f n'est pas symétrique, on effectue une symétrie centrale avant application de la convolution

1. Rappels

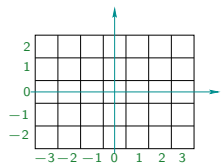
1.1. Définition

$$(f * g)(i,j) = \sum_{i'=-D_i}^{D_i} \sum_{j'=-D_j}^{D_j} f(i-i', j-j')\, g(i',j')$$

- pour chaque pixel P : $\begin{cases} \text{translation du symétrique de } f \text{ sur } P \\ \text{somme des produits terme à terme} \end{cases}$
- en pratique, le noyau f est une matrice rectangulaire de petite taille

$$(2D_i + 1) \times (2D_j + 1)$$

exemple pour $D_i = 2$ et $D_j = 3$:



- si le noyau f n'est pas symétrique, on effectue une symétrie centrale avant application de la convolution

1. Rappels

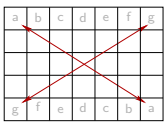
1.1. Définition

$$(f * g)(i,j) = \sum_{i'=-D_i}^{D_i} \sum_{j'=-D_j}^{D_j} f(i-i', j-j')\, g(i',j')$$

- pour chaque pixel P : $\begin{cases} \text{translation du symétrique de } f \text{ sur } P \\ \text{somme des produits terme à terme} \end{cases}$
- en pratique, le noyau f est une matrice rectangulaire de petite taille

$$(2D_i + 1) \times (2D_j + 1)$$

exemple pour $D_i = 2$ et $D_j = 3$:



- si le noyau f n'est pas symétrique, on effectue une symétrie centrale avant application de la convolution

1.2. Exemples de noyaux

- coefficients positifs : moyenne, lissage, etc.

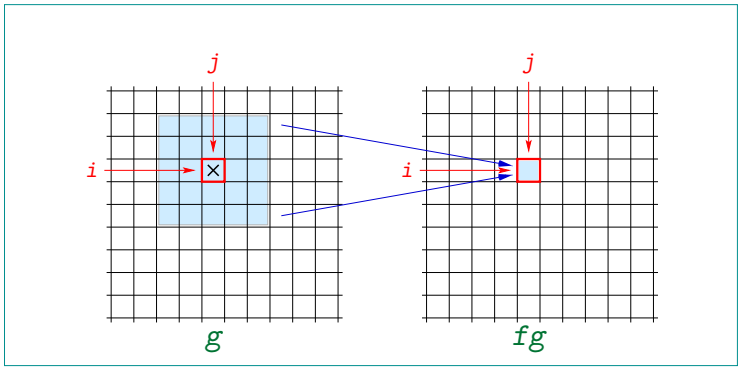
$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

- coefficients positifs et négatifs : approximation de dérivées par différences finies (premières, secondes...), contours, rehaussement de détails, etc.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -5 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2. Mise en œuvre

2.1. Algorithme

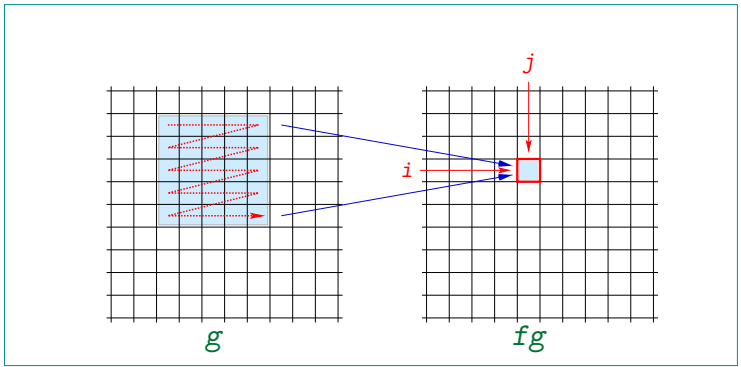



```
fg[i][j] = 0;
for (int ip = 0; ip < 2*Di+1; ip++)
    for (int jp = 0; jp < 2*Dj+1; jp++)
        fg[i][j] += g[i - Di + ip][j - Dj + jp] * f[ip][jp]
```

- seule difficulté : traitement des bords de l'image

2. Mise en œuvre

2.1. Algorithme

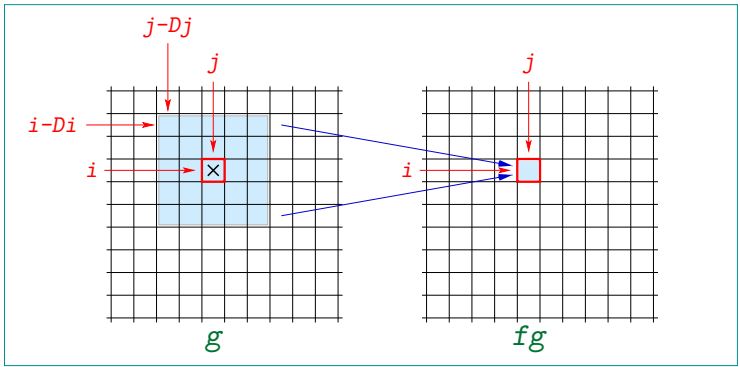



```
fg[i][j] = 0;
for (int ip = 0; ip < 2*Di+1; ip++)
    for (int jp = 0; jp < 2*Dj+1; jp++)
        fg[i][j] += g[i - Di + ip][j - Dj + jp] * f[ip][jp]
```

- seule difficulté : traitement des bords de l'image

2. Mise en œuvre

2.1. Algorithme

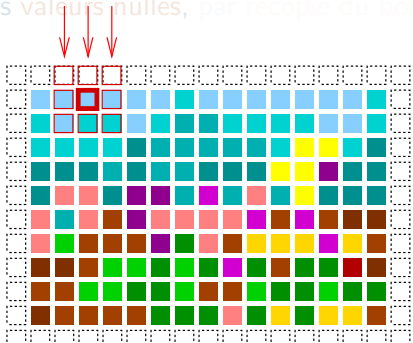



```
fg[i][j] = 0;
for (int ip = 0; ip < 2*Di+1; ip++)
    for (int jp = 0; jp < 2*Dj+1; jp++)
        fg[i][j] += g[i - Di + ip][j - Dj + jp] * f[ip][jp]
```

- seule difficulté : traitement des bords de l'image

2.2. Traitement des bords de l'image

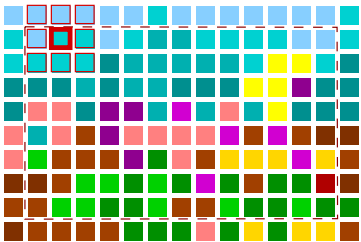
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples sont suffisantes

2.2. Traitement des bords de l'image

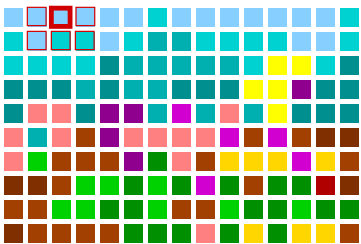
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples dont suffisantes

2.2. Traitement des bords de l'image

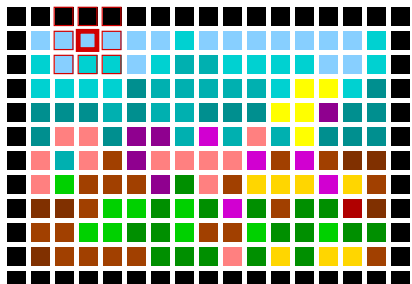
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples dont suffisantes

2.2. Traitement des bords de l'image

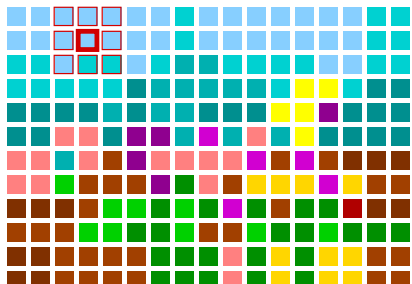
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples sont suffisantes

2.2. Traitement des bords de l'image

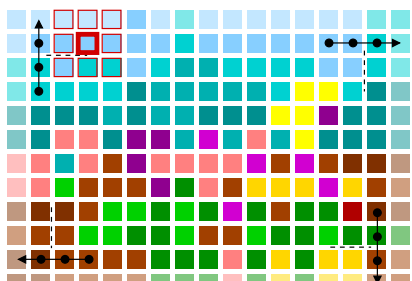
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples sont suffisantes

2.2. Traitement des bords de l'image

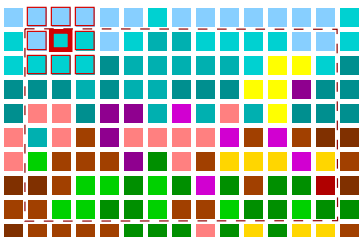
- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples dont suffisantes

2.2. Traitement des bords de l'image

- on ignore les pixels du bord :
  bande de largeur D_i (verticalement) ou D_j (horizontalement)
- on restreint le calcul au domaine valide : renormalisation du noyau
- on prolonge l'image par des valeurs nulles, par recopie du bord, par extrapolation...



- en général, les méthodes les plus simples dont suffisantes

3. Optimisation de la mise en œuvre

3.1. Opérateurs séparables

- coût de la mise en œuvre : croissance polynomiale selon la taille du noyau
- certains opérateurs de convolution sont séparables :

  décomposition de la convolution 2D en deux convolutions 1D

$$f_{xy} = f_x * f_y$$

- application de l'associativité de $*$

$$\begin{aligned} f_{xy} * g &= (f_x * f_y) * g \\ &= f_x * (f_y * g) \end{aligned}$$

- complexité : $N^2 \to 2N$

3. Optimisation de la mise en œuvre

3.1. Opérateurs séparables

- coût de la mise en œuvre : croissance polynomiale selon la taille du noyau
- certains opérateurs de convolution sont séparables :

  décomposition de la convolution 2D en deux convolutions 1D

$$f_{xy} = f_x * f_y$$

- application de l'associativité de $*$

$$\begin{aligned} f_{xy} * g &= (f_x * f_y) * g \\ &= f_x * (f_y * g) \end{aligned}$$

- complexité : $N^2 \to 2N$

- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
0	0	0

$*$

$$\begin{matrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{matrix}$$

- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

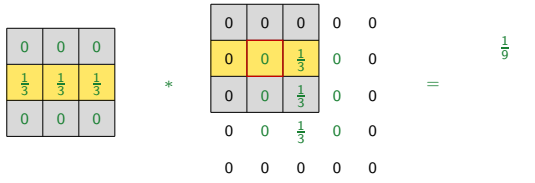
0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
0	0	0

$*$

0	0	0	0	0
0	0	$\frac{1}{3}$	0	0
0	0	$\frac{1}{3}$	0	0
0	0	$\frac{1}{3}$	0	0
0	0	0	0	0

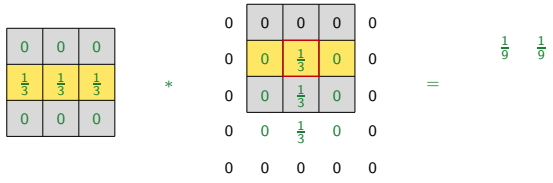
- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



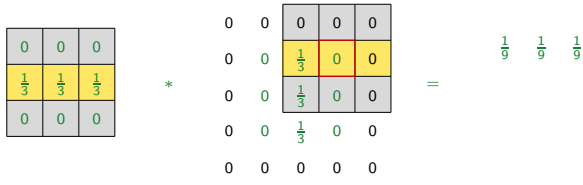
- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



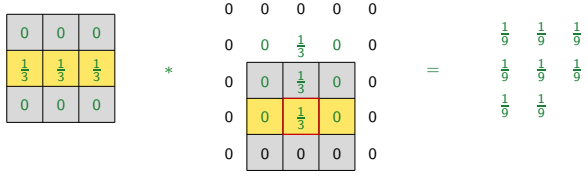
- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



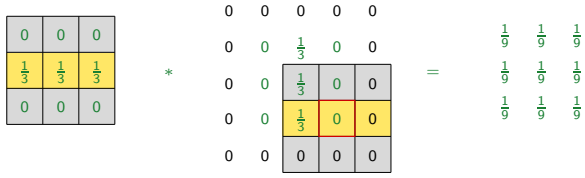
- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



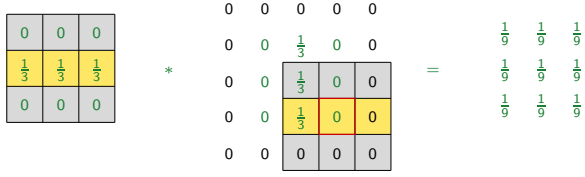
- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



- exemple : filtre boîte (moyenne)

$$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



- en pratique : composition de deux convolutions 1D

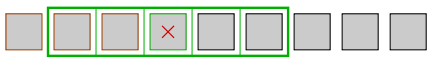
$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} * g \qquad \text{et} \qquad \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} * g$$

3.2. Filtrage récursif

- certains filtres séparables admettent une implémentation récursive : coût indépendant de la taille du noyau
- en 1D, on note

  $P - i$: le $i^{\text{ème}}$ voisin gauche de P
  $P + i$: le $i^{\text{ème}}$ voisin droit de P

$$I'(P) = a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots + b_1 I(P-1) + b_2 I(P-2) + \dots$$



- on réutilise les valeurs déjà calculées : $I'(P-i)$

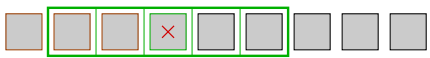
$$I'(P) = a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots + b_1 I(P-1) + b_2 I(P-2) + \dots + b_1 I'(P-1) + b_2 I'(P-2) + \dots$$

3.2. Filtrage récursif

- certains filtres séparables admettent une implémentation récursive : coût indépendant de la taille du noyau
- en 1D, on note

  $P - i$: le $i^{\text{ème}}$ voisin gauche de P
  $P + i$: le $i^{\text{ème}}$ voisin droit de P

$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \end{aligned}$$



- on réutilise les valeurs déjà calculées : $I'(P-i)$

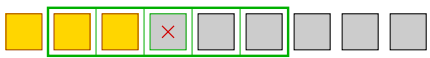
$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \\ &\quad + b'_1 I'(P-1) + b'_2 I'(P-2) + \dots \end{aligned}$$

3.2. Filtrage récursif

- certains filtres séparables admettent une implémentation récursive : coût indépendant de la taille du noyau
- en 1D, on note

  $P - i$: le $i^{\text{ème}}$ voisin gauche de P
  $P + i$: le $i^{\text{ème}}$ voisin droit de P

$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \end{aligned}$$



- on réutilise les valeurs déjà calculées : $I'(P-i)$

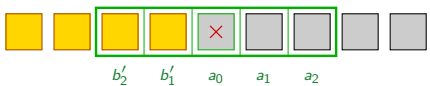
$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \\ &\quad + b'_1 I'(P-1) + b'_2 I'(P-2) + \dots \end{aligned}$$

3.2. Filtrage récursif

- certains filtres séparables admettent une implémentation récursive : coût indépendant de la taille du noyau
- en 1D, on note

  $P - i$: le $i^{\text{ème}}$ voisin gauche de P
  $P + i$: le $i^{\text{ème}}$ voisin droit de P

$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \end{aligned}$$



- on réutilise les valeurs déjà calculées : $I'(P-i)$

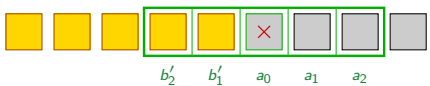
$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \\ &\quad + b'_1 I'(P-1) + b'_2 I'(P-2) + \dots \end{aligned}$$

3.2. Filtrage récursif

- certains filtres séparables admettent une implémentation récursive : coût indépendant de la taille du noyau
- en 1D, on note

  $P - i$: le $i^{\text{ème}}$ voisin gauche de P
  $P + i$: le $i^{\text{ème}}$ voisin droit de P

$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \end{aligned}$$



- on réutilise les valeurs déjà calculées : $I'(P-i)$

$$\begin{aligned} I'(P) &= a_0 I(P) + a_1 I(P+1) + a_2 I(P+2) + \dots \\ &\quad + b_1 I(P-1) + b_2 I(P-2) + \dots \\ &\quad + b'_1 I'(P-1) + b'_2 I'(P-2) + \dots \end{aligned}$$

○ Exemple : filtre boîte de demi-largeur 3

$$\left(\frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \right)$$

- au point $P-1$:

$$\begin{aligned} I'(P-1) &= \frac{1}{7}\Big(I(P-4)+I(P-3)+I(P-2)+I(P-1) \\ &\quad +I(P)+I(P+1)+I(P+2)\Big) \end{aligned}$$

- au point P :

$$\begin{aligned} I'(P) &= \frac{1}{7}\Big(I(P-3)+I(P-2)+I(P-1)+I(P) \\ &\quad +I(P+1)+I(P+2)+I(P+3)\Big) \end{aligned}$$

- calcul récursif de $I'(P)$ en fonction de $I'(P-1)$

$$I'(P) = I'(P-1) + \frac{1}{7}\Big(I(P+3) - I(P-4)\Big)$$

- généralisable : coût constant quelle que soit la taille du noyau

○ Exemple : filtre boîte de demi-largeur 3

$$\left(\frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \right)$$

- au point $P-1$:

$$\begin{aligned} I'(P-1) &= \frac{1}{7}\Big(I(P-4)+I(P-3)+I(P-2)+I(P-1) \\ &\quad +I(P)+I(P+1)+I(P+2)\Big) \end{aligned}$$

- au point P :

$$\begin{aligned} I'(P) &= \frac{1}{7}\Big(I(P-3)+I(P-2)+I(P-1)+I(P) \\ &\quad +I(P+1)+I(P+2)+I(P+3)\Big) \end{aligned}$$

- calcul récursif de $I'(P)$ en fonction de $I'(P-1)$

$$I'(P) = I'(P-1) + \frac{1}{7}\Big(I(P+3) - I(P-4)\Big)$$

- généralisable : coût constant quelle que soit la taille du noyau

○ Exemple : filtre boîte de demi-largeur 3

$$\left(\frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \right)$$

- au point $P-1$:

$$\begin{aligned} I'(P-1) &= \frac{1}{7}\Big(I(P-4)+I(P-3)+I(P-2)+I(P-1) \\ &\quad +I(P)+I(P+1)+I(P+2)\Big) \end{aligned}$$

- au point P :

$$\begin{aligned} I'(P) &= \frac{1}{7}\Big(I(P-3)+I(P-2)+I(P-1)+I(P) \\ &\quad +I(P+1)+I(P+2)+I(P+3)\Big) \end{aligned}$$

- calcul récursif de $I'(P)$ en fonction de $I'(P-1)$

$$I'(P) = I'(P-1) + \frac{1}{7}\Big(I(P+3) - I(P-4)\Big)$$

- généralisable : coût constant quelle que soit la taille du noyau

○ Exemple : filtre boîte de demi-largeur 3

$$\left(\frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \right)$$

- au point $P-1$:

$$\begin{aligned} I'(P-1) &= \frac{1}{7}\Big(I(P-4)+I(P-3)+I(P-2)+I(P-1) \\ &\quad +I(P)+I(P+1)+I(P+2)\Big) \end{aligned}$$

- au point P :

$$\begin{aligned} I'(P) &= \frac{1}{7}\Big(I(P-3)+I(P-2)+I(P-1)+I(P) \\ &\quad +I(P+1)+I(P+2)+I(P+3)\Big) \end{aligned}$$

- calcul récursif de $I'(P)$ en fonction de $I'(P-1)$

$$I'(P) = I'(P-1) + \tfrac{1}{7}\Big(I(P+3) - I(P-4)\Big)$$

- généralisable : coût constant quelle que soit la taille du noyau

○ Exemple : filtre boîte de demi-largeur 3

$$\left(\frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \quad \frac{1}{7} \right)$$

- au point $P-1$:

$$\begin{aligned} I'(P-1) &= \frac{1}{7}\Big(I(P-4)+I(P-3)+I(P-2)+I(P-1) \\ &\quad +I(P)+I(P+1)+I(P+2)\Big) \end{aligned}$$

- au point P :

$$\begin{aligned} I'(P) &= \frac{1}{7}\Big(I(P-3)+I(P-2)+I(P-1)+I(P) \\ &\quad +I(P+1)+I(P+2)+I(P+3)\Big) \end{aligned}$$

- calcul récursif de $I'(P)$ en fonction de $I'(P-1)$

$$I'(P) = I'(P-1) + \frac{1}{7}\Big(I(P+3)-I(P-4)\Big)$$

- généralisable : coût constant quelle que soit la taille du noyau