Comment le sens se traduit en pixel?

Il est très compliqué de passer d'un tas de nombres associés à des pixels jusqu'à la détection d'un objet.

Comment le sens se traduit en pixel?

Il est très compliqué de passer d'un tas de nombres associés à des pixels jusqu'à la détection d'un objet.

D'après vous, quelles sont les facteurs de variations?

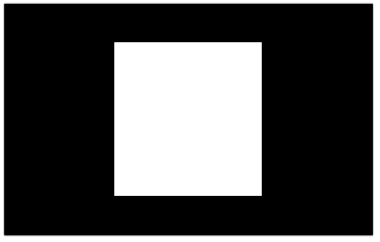
Comment le sens se traduit en pixel?

Il est très compliqué de passer d'un tas de nombres associés à des pixels jusqu'à la détection d'un objet.

L'apparence d'un objet peut varier par:

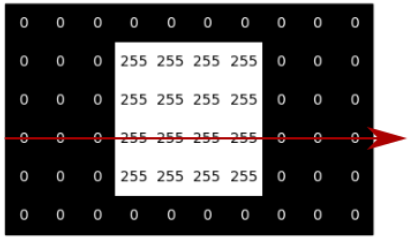
- Translation
- Rotation
- Point de vue
- Changement dans l'illumination
- variation intra-classe

Detection de contours



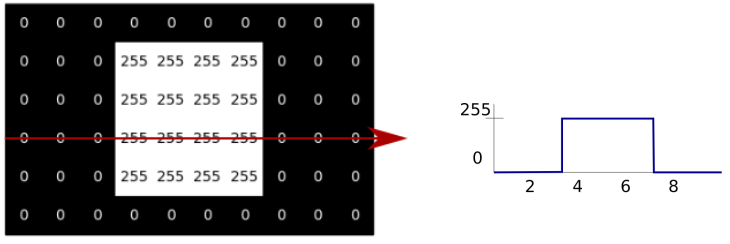
Commençons donc: supposons que nous souhaitons détecter les contours sur cette image.

Detection de contours



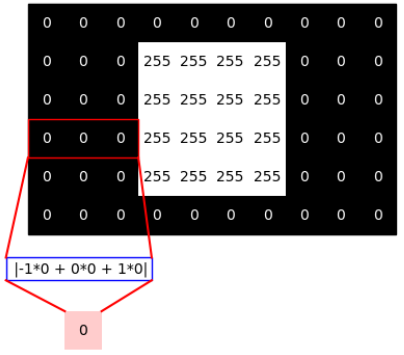
En particulier sur cette ligne

Detection de contours



Les contours apparaissent aux changements brusques de valeur des pixels.

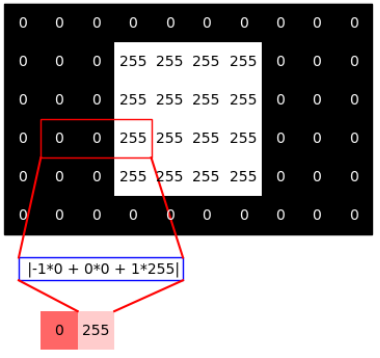
Détection de contours



Appliquons un filtre k sur une des lignes avec:

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

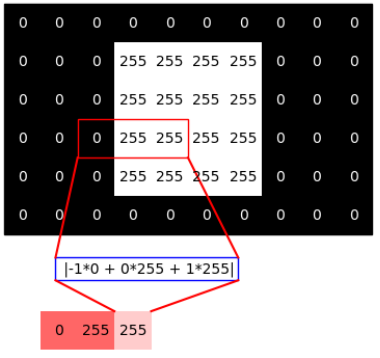
Détection de contours



Appliquons un filtre k sur une des lignes avec:

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

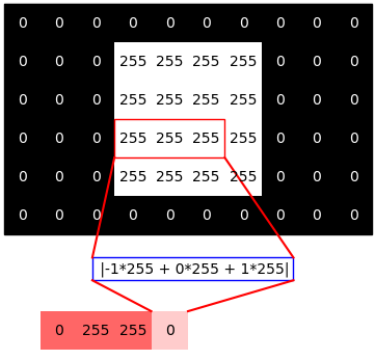
Détection de contours



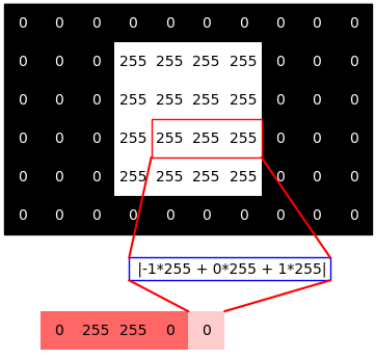
Appliquons un filtre k sur une des lignes avec:

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

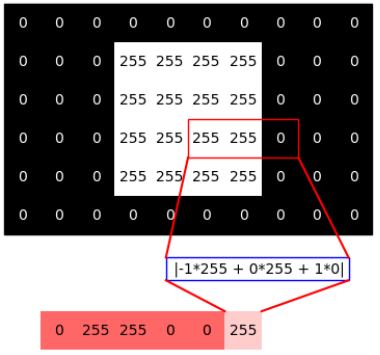
Détection de contours



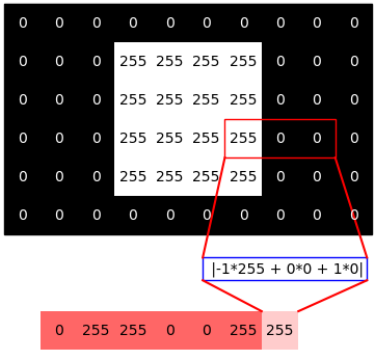
Détection de contours



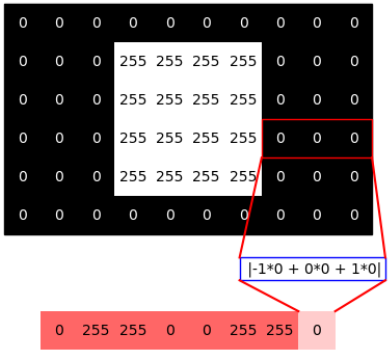
Détection de contours



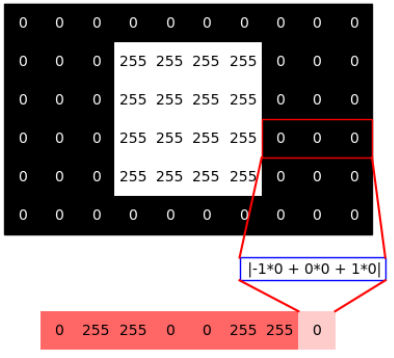
Détection de contours

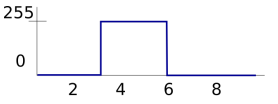


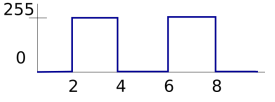
Détection de contours



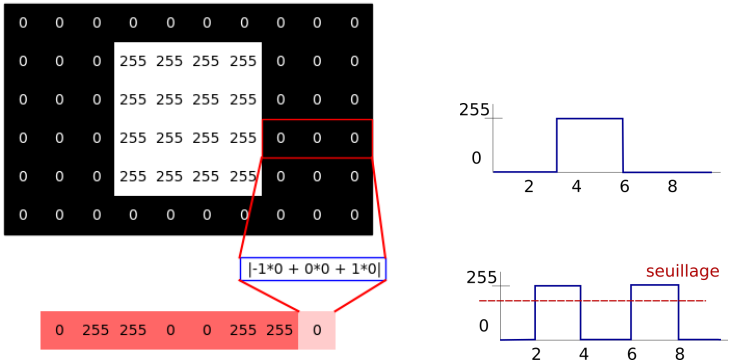
Détection de contours







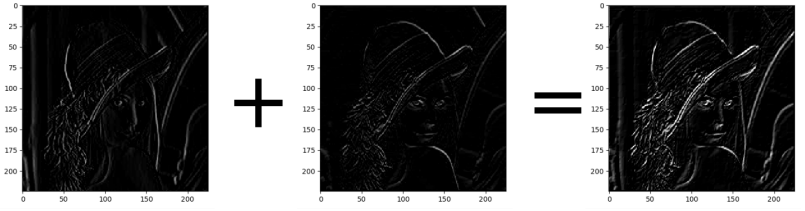
Détection de contours



Convolution 1D discrète entre un signal f et un filtre k de taille d :

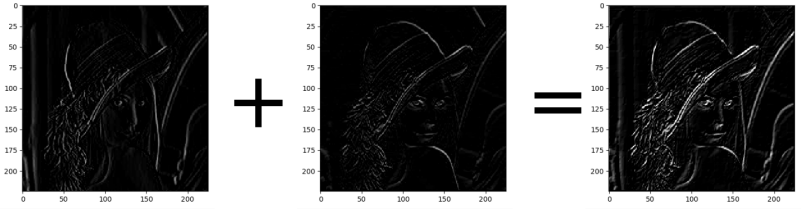
$$(f \star k)(i) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n).k(n)$$

Application sur une image réelle



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

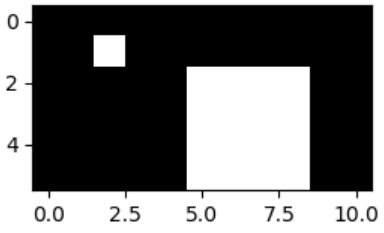
Application sur une image réelle



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

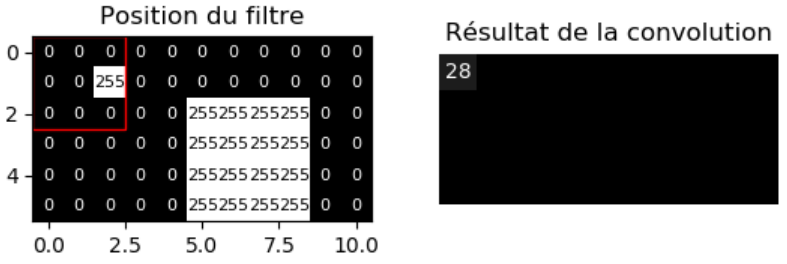
Notez que les contours ne correspondent pas exactement aux frontières des objets.

Filtre moyenneur: convolution en 2D



Voyons un traitement pour retirer les artefacts et adoucir les contours d'une image.

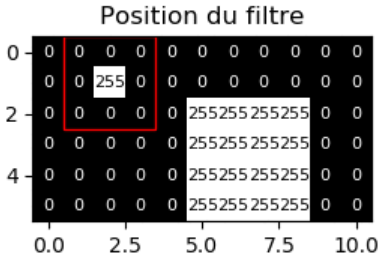
Filtre moyenneur: convolution en 2D

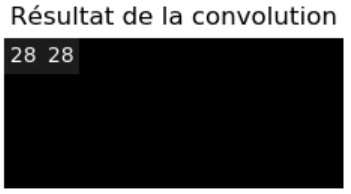


Convolution avec un filtre Moyenneur K

```
In [6]: K = np.ones((3,3)) / 9
array([[0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1]])
```

Filtre moyenneur: convolution en 2D

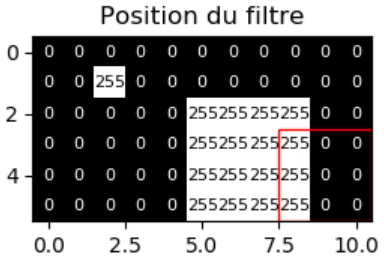




Convolution avec un filtre Moyenneur K

```
In [6]: K = np.ones((3,3)) / 9
array([[0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1]])
```

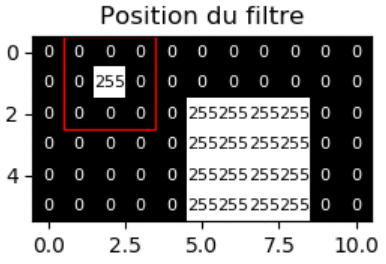
Filtre moyenneur: convolution en 2D

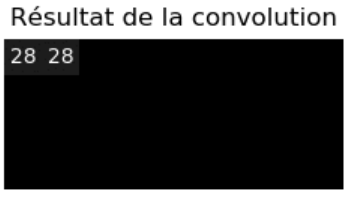


Convolution avec un filtre Moyenneur K

```
In [6]: K = np.ones((3,3)) / 9
array([[0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1],
       [0.1, 0.1, 0.1]])
```

Filtre moyenneur: convolution en 2D





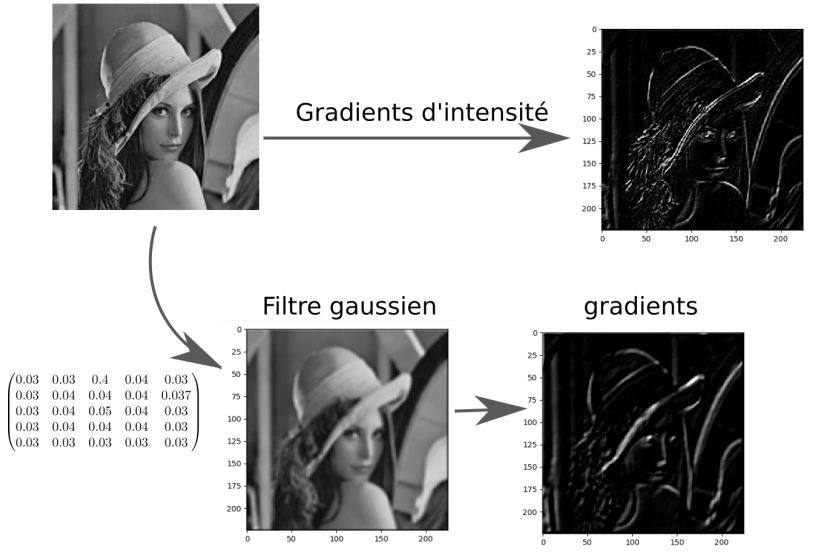
$$(img \star k)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} img(i-n, j-m).k(n,m)$$

ou en python:

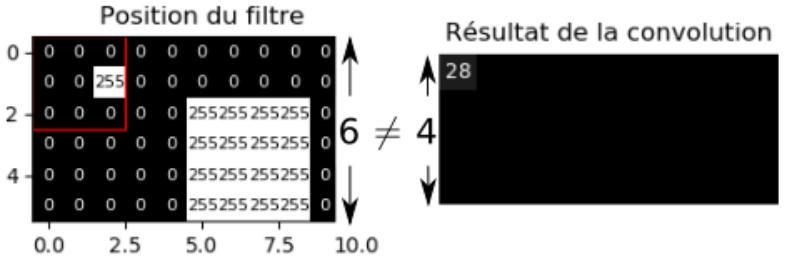
```
# img[i-(d-1)/2:i+(d-1)/2,j-(d-1)/2:j+(d-1)/2,] pour cropper une partie de l'image à la posi
# sum.() somme toutes les valeurs du tableau
result_convolution[i,j] = (k * img[i-(d-1)/2:i+(d-1)/2,j-(d-1)/2:j+(d-1)/2,]).sum()
```

Application sur une image réelle

Avec un filtre gaussien 5 x 5



Effet de bord



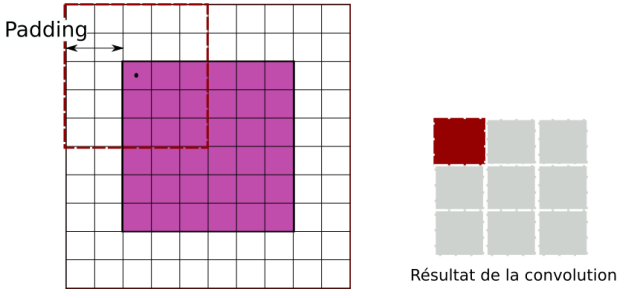
-Problème : comment traiter le pixel en (0,0)?

-Solution couramment utilisée:

padding with zéros, ou la valeur des pixels voisins.

Convolution 2D en général

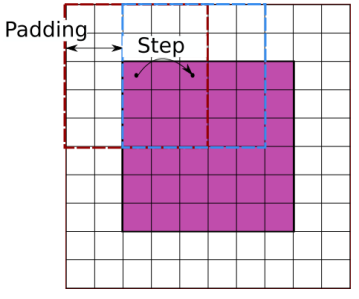
Filtre de taille $(f_w, f_h) = (5, 5)$, sur une image $(I_w, I_h) = (6, 6)$, avec un padding $p = 2$ et un pas $s = 2$.

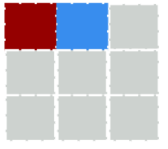


Résultat de la convolution

Convolution 2D en général

Filtre de taille $(f_w, f_h) = (5, 5)$, sur une image $(I_w, I_h) = (6, 6)$, avec un padding $p = 2$ et un pas $s = 2$.

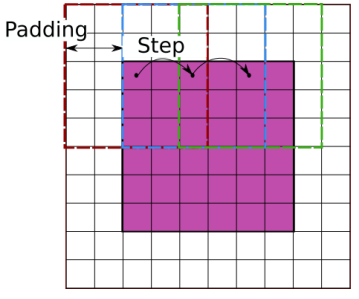




Résultat de la convolution

Convolution 2D en général

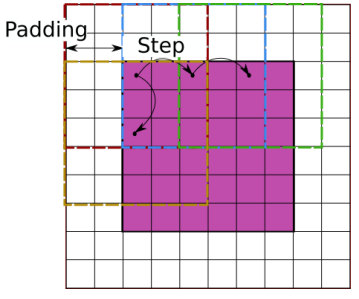
Filtre de taille $(f_w, f_h) = (5, 5)$, sur une image $(I_w, I_h) = (6, 6)$, avec un padding $p = 2$ et un pas $s = 2$.



Résultat de la convolution

Convolution 2D en général

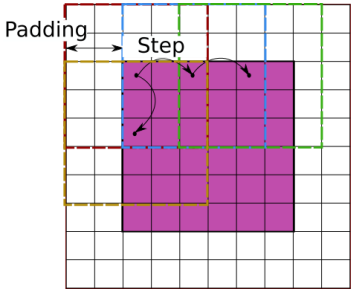
Filtre de taille $(f_w, f_h) = (5, 5)$, sur une image $(I_w, I_h) = (6, 6)$, avec un padding $p = 2$ et un pas $s = 2$.

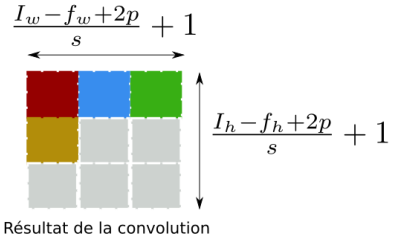


Résultat de la convolution

Convolution 2D en général

Filtre de taille $(f_w, f_h) = (5, 5)$, sur une image $(I_w, I_h) = (6, 6)$, avec un padding $p = 2$ et un pas $s = 2$.



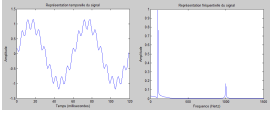


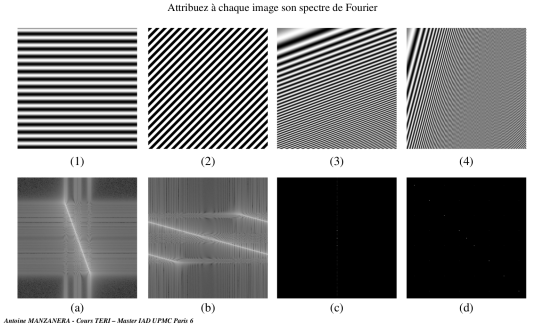

```
# initialisation of the convolution result
res = np.zeros((math.ceil( (h - fh + 2*p) /s + 1), math.ceil((w - fw + 2*p)/s + 1)))
```

Autres convolutions célébres

Fourier, avec $k(x, \xi) = e^{-i\xi x}$

$$(f \star k)(\xi) = \int_{-\inf}^{\inf} f(x) e^{-i\xi x} dx$$





Lab : Implémentation de convolutions

Ouvrir le fichier: TP*convolutions.ipynb*

Faire l'exercice 2