

TP1 : Détection de contours (ImageJ/Matlab)

IUT Cachan

N. Gac

1 L'opérateur Gradient

Un contour est caractérisé par un changement brutal de valeur de pixels voisins. Les techniques les plus simples et les plus efficaces de détection de contours utilisent le gradient G de l'image, correspondant aux dérivées premières de l'image selon les directions x et y . Dans une image numérisée, la dérivée première au niveau d'un pixel (Dx et Dy) s'approxime par la différence entre les valeurs des deux pixels voisins.

$$G = \sqrt{Dx^2 + Dy^2}$$

$$\begin{cases} Dx = E - O \\ Dy = N - S \end{cases}$$

	N	
O	C	E
	S	



(a) Image originale



(b) Gradient

Figure 1: Détection de contours

2 Détection de contours d'images bruitées sous ImageJ

Vous effectuerez tout d'abord la détection de contours à l'aide d'ImageJ, logiciel de traitement d'image simple d'utilisation. Vous travaillerez sur deux images : un carre blanc et une ferrari. Cette détection se fera avec ou sans bruit (voir figure 5).

2.1 Images sans bruit

a) Ouvrez l'image *carre.raw*. C'est une image au format raw (512*512 float 32 bits), c'est à dire au format brut. Les données sont stockées en format binaire (sans entête indiquant la largeur et la hauteur de l'image, le format des données). Faites *File->Import->Raw*, indiquez hauteur et largeur de 512, une *Image type* de *32 bit Real*. Cochez la case *little endian*

b) Afin de faire ressortir les contours verticaux, nous allons appliquer un simple opérateur correspondant à $|Dx|$ grâce aux filtres d'ImageJ :

- Dérivée selon x : *Process->Filters->Convolve* avec le filtre -1 0 1
- Valeur absolue : *Process->Maths->abs*

c) Observez un profil (ou coupe) de cette image grace à *Analyse->Plot Profile* après avoir sélectionné une ligne horizontale de pixels.

d) Faites la même étude pour l'image 960*1280 *ferrari_NB.raw*.

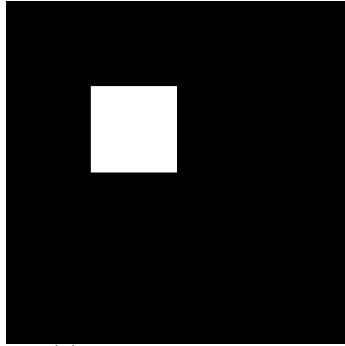
2.2 Images avec ajout de bruit gaussien

Bien souvent du bruit vient entâcher l'image et rend plus difficile la détection de contours. C'est le cas de l'image *carre_noise_gaussian.raw*.

a) Comparez tout d'abord les profils des deux images avec et sans bruit. Puis appliquez l'opérateur $|Dx|$ de détection de contours verticaux sur l'image bruitée. Qu'observez vous ?

b) Une technique simple de débruitage de l'image comme pré-étape à la détection de contours consiste à appliquer un filtre moyennneur. Un pixel est remplacé par la moyenne sur son entourage. Appliquez le filtre moyennneur d'ImageJ (*Process->Filter->Mean*) sur l'image bruitée puis appliquez l'opérateur de détection de contours verticaux.

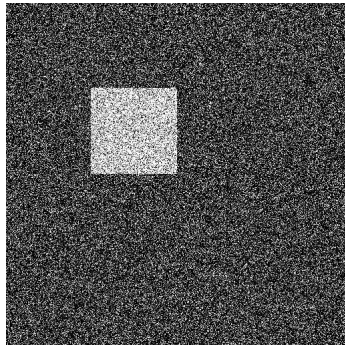
c) Faites la même étude pour l'image 960*1280 *ferrari_NB_bruit_gaussien.raw*.



(a) Carré blanc original



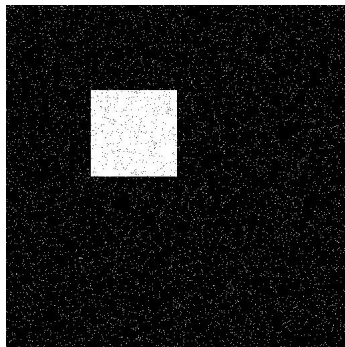
(b) Ferrari d'origine



(c) Carré blanc avec bruit gaussien



(d) Ferrari avec bruit gaussien sur la carrosserie



(e) Carré blanc avec bruit poivre et sel



(f) Ferrari avec bruit poivre et sel sur la carrosserie

Figure 2: Images sans bruit, avec bruit gaussien et bruit poivre et sel

2.3 Images avec ajout de bruit poivre et sel

Le bruit peut être de nature différent que du bruit gaussien comme par exemple du bruit poivre et sel correspondant à du bruit impulsionnel.

a) Appliquez le filtre moyenneur puis l'opérateur de détection de contours verticaux sur l'image *carre_noise_seltpepper.raw*. Etes vous satisfait du résultat ?

b) Une technique plus adaptée que le filtre moyenneur comme débruiteur est d'utiliser un filtre median. Un pixel est remplacé par la valeur médiane sur son entourage. Utilisez le filtre median d'Image J pour améliorer la détection de contours (*Process->Filter->Median*).

c) Faites la même étude pour l'image 960*1280 *ferrari_NB_bruit_poivre_et_sel.raw*.

3 Convolution 2D en Matlab

3.1 Convolution 2D : $g=f*h$

Les opérateurs de dérivée première et de moyennage/lissage sont linéaires et correspondent à des convolutions 2D. Nous rappelons ci-dessous l'expression de la convolution d'une fonction discrete f à une dimension avec un filtre h (voir figure 3).

$$f_{conv}(n) = (f * h)(n) = \sum_{m=-\infty}^{\infty} f(m)h(n-m)$$

Avec $m=n+k$, nous obtenons :

$$f_{conv}(n) = (f * h)(n) = \sum_{k=-\infty}^{\infty} f(n+k)h(-k)$$

Pour un filtre avec un nombre impair de coefficients et de rayon R_h , nous avons :

$$f_{conv}(n) = (f * h)(n) = \sum_{k=-R_h}^{R_h} f(n+k)h(-k)$$

En 2D, la convolution d'une image I avec un filtre 2D h a alors pour expression :

$$I_{conv}(xn, yn) = (I * h)(xn, yn) = \sum_{ky=-R_h}^{R_h} \sum_{kx=-R_h}^{R_h} f(xn+kx, yn+ky)h(-kx, -ky)$$

Donnez l'expression des filtres 2D h , correspondant aux opérateurs D_x , D_y et de lissage (moyenne sur tous les voisins d'un pixel).

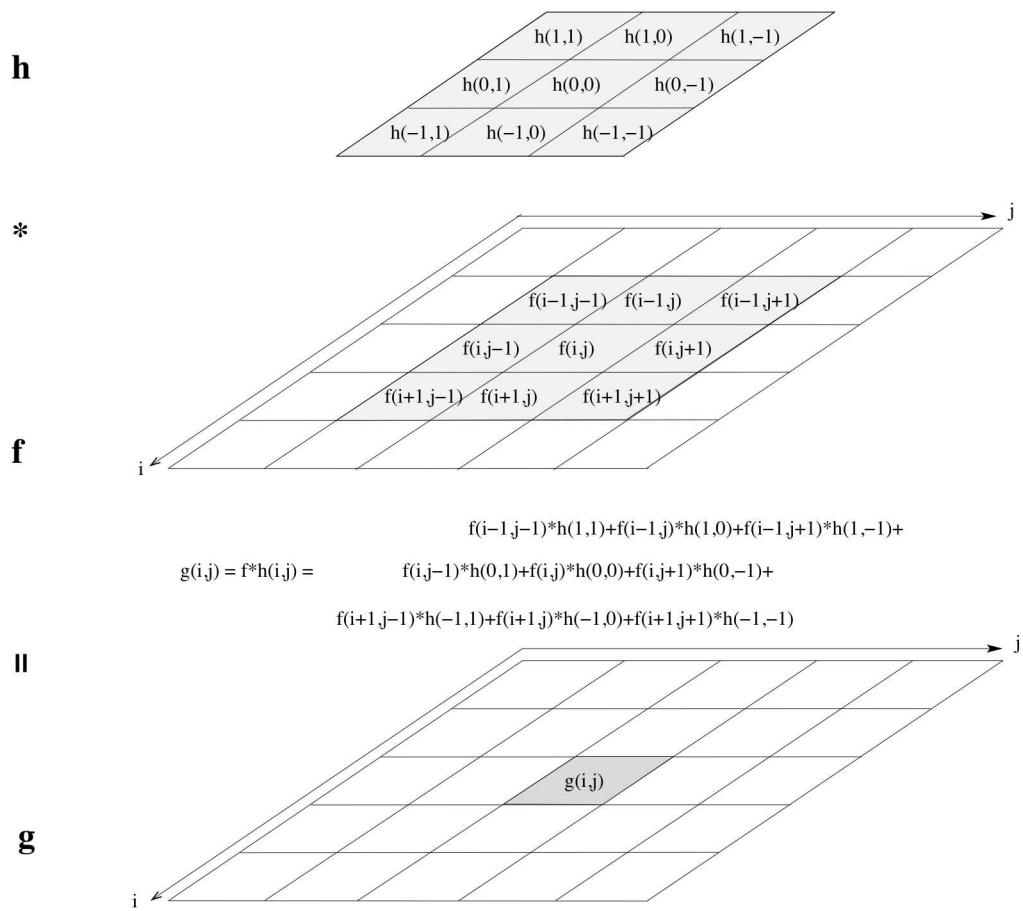


Figure 3: Convolution 2D

3.2 Matlab

3.2.1 Filtre avec conv2

Effectuez sous Matlab, la détection de contours des images *ferrari_NB.raw*, *ferrari_NB_bruit_gaussien.raw*, *ferrari_NB_bruit_poivre_et_sel.raw*. Vous utiliserez la fonction de convolution 2D, *conv2* pour calculer le gradient (Dx et Dy). Pour le débruitage gaussien, vous pouvez utiliser un filtre moyenneur rectangulaire puis gaussien toujours avec *conv2*. Pour le filtre median, vous pouvez utiliser la fonction *medfilt2*.

Vous utiliserez comme script de départ le fichier *Contours_Matlab.m*.

Faites varier la taille des filtres et observer les résultats.

3.2.2 Qu'est ce qu'un un filtre gaussien et comment le créer sous matlab ?

En probabilité, une distribution gaussienne 1D a pour expression :

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{(x - \mu)^2}{2\sigma^2}$$

avec μ la moyenne et σ l'écart type

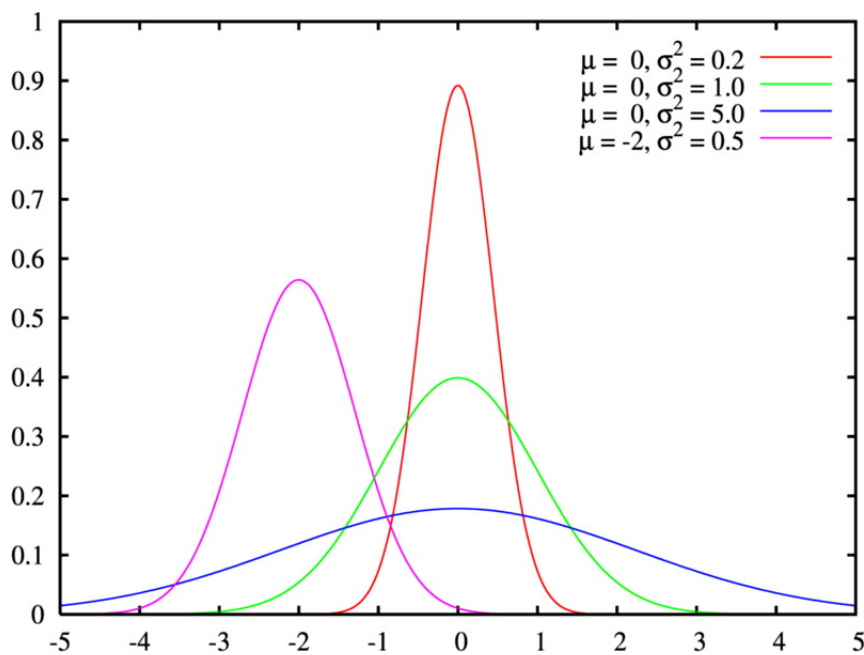


Figure 4: Fonction gaussienne 1D

En 2D, l'expression devient :

$$p(x, y) = A \cdot \exp - \left(\frac{(x - \mu_x)^2}{2\sigma_x^2} + \frac{(y - \mu_y)^2}{2\sigma_y^2} \right)$$

avec μ_x et μ_y les moyennes en x et y, σ_x et σ_y les écarts type et A, le facteur de normalisation ($\int \int p(x, y) dx dy = 1$)

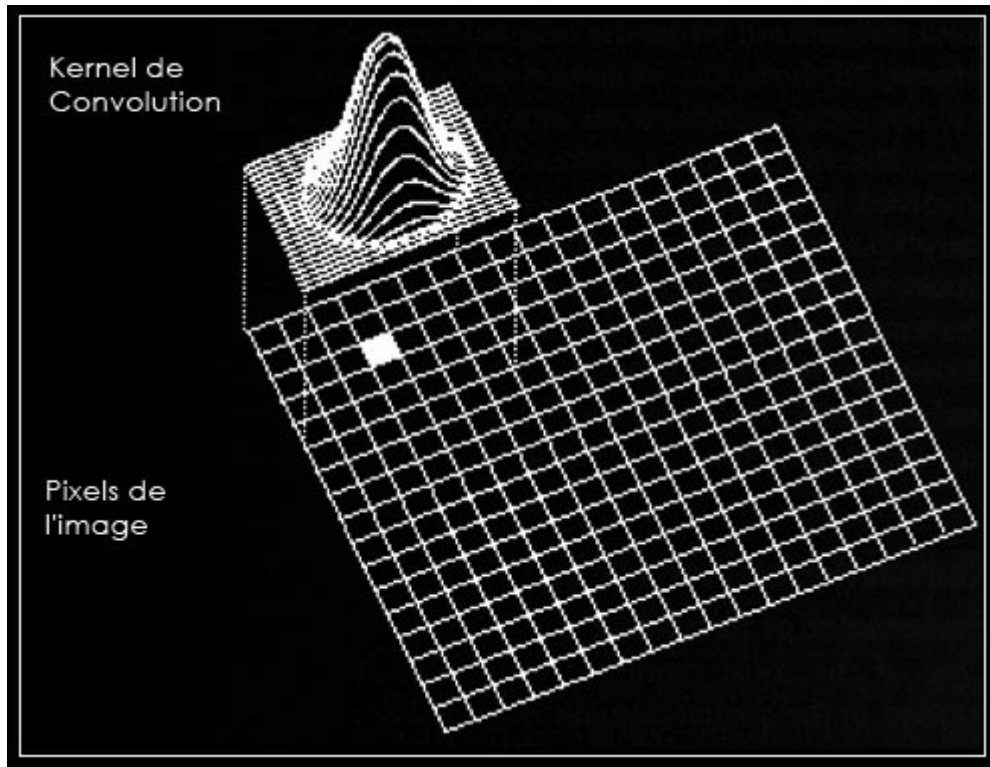


Figure 5: Filtre gaussien 2D

Sous matlab, pour créer votre filtre gaussien 2D vous utiliserez la fonction **meshgrid** pour créer les tableaux 2D X et Y contenant respectivement les coordonnées x et y des "cases" du filtre. Utilisez l'aide de Matlab. Pensez à normaliser votre filtre 2D, autrement dit faites en sorte que la somme des coefficients du filtre soit égale à 1.