TP2 : Traitement d'images Numériques

Filtrage spatial et fréquentiel

Matière : Imagerie embarquée — Niveau : IIA4

Enseignant responsable : Mme Ben Saïd Salma

L'objectif de ce TP est de manipuler les différents outils de filtrage d'images numériques :
Dans le domaine spatial :

- Filtrage par différents noyaux de convolution, lissage, réduction de bruit et détection de contours.
- Filtrage non linéaire.
- Filtrage morphologique.

Dans le domaine spectral :

- Compréhension et application de la transformée de Fourier bi-dimensionnelle.
- Filtrage fréquentiel.

I. Filtrage dans le domaine spatial

Le filtrage spatial consiste à réaliser un produit de convolution entre une image source et un filtre appelé noyau. Le noyau doit être impair pour assurer la symétrie. Ceci, est réalisé par la fonction suivante de OpenCV:

resulting_image = cv2.filter2D(src, ddepth, kernel)

ddepth: profondeur de l'image destination. La **Valeur -1 pour avoir la même profondeur que l'image** source.
Le résultat du filtrage va dépendre des valeurs des coefficients du noyau (le kernel).

A. Filtrage passe bas

- Appliquer les différents noyaux de convolution passe bas ci-dessous sur l'image **circuit.tif**

$\frac{1}{81}$

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

pyramidal

$\frac{1}{256}$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

binomial

$\frac{1}{25}$

0	0	1	0	0
0	2	2	2	0
1	2	5	2	1
0	2	2	2	0
0	0	1	0	0

conique

Appliquer le filtre gaussien de la fonction ***cv2.GaussianBlur(src, (5,5), 0) .***
La qualité du résultat de filtrage peut être évaluée par des métriques tels que :

Signal-to-Noise Ratio

$$SNR = 10\log_{10}\left(\frac{\sum I^2}{\sum (I - I_f)^2}\right)$$

Peak Signal-to-Noise Ratio

$$PSNR = 10 \cdot \log_{10}\left(\frac{L^2}{MSE}\right)$$

Structural Similarity Index

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Calculer le PSNR et SSIM, afficher l'image originale, les images filtrées et les résultats obtenus.

B. Filtrage passe haut : détection de contours

- Appliquer le noyau de convolution de prewitt sur l'image **circuit.tif** non filtrée puis sur cette même image filtrée
  Afficher et interpréter les résultats obtenus. Utiliser la même ligne de profil sur les différents résultats pour faciliter l'interprétation.

- Construire une image binaire nommée I comportant un rectangle.
  Construire les noyaux de convolution(ligne/colonne) prewitt et sobel. Détecter le contour du rectangle en utilisant un filtrage spatial sur l'image à l'aide de chacun des noyaux construits, Afficher, constater.
  Changer dans la fonction cv2.filter2D() le depth de -1 à cv2.CV_64F, puis refaire le filtrage. Afficher, Interpréter.
  Appliquer le filtre de canny en utilisant la fonction prédéfinie d'OpenCV :
  img_canny = cv2.Canny(I,threshmin,threshmax)
- Charger l'image gantrycrane.png, appliquer le filtrage précédent sur cette image. Afficher.

C. Filtrage non linéaire : élimination du bruit impulsionnel

Charger l'image cellSP.tif, Cette image contient un bruit de type sel et poivre.
Réaliser un filtrage moyenneur et un filtrage médian sur l'image en utilisant les fonctions suivantes :

cv2.blur(img,(n,n)) et **cv2.medianBlur(img,n),** `n` représente la taille du noyau.

Afficher et interpréter.

D. Filtrage Morphologique

OpenCV présente les fonctions suivantes respectivement pour l'érosion, la dilatation, l'ouverture et la fermeture :

```
cv2.erode(img,kernel,iterations = n),
cv2.dilate(img,kernel,iterations = n),
cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel),
cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel).
```

Toute transformation morphologique utilise un élément structurant et dont le résultat de la transformation dépend de sa forme et de sa taille. Cet élément structurant est défini par la fonction :

```
kernel = cv2.getStructuringElement(forme,taille)
```

Exemple de forme : `cv2.MORPH_RECT, cv2.MORPH_CROSS, cv2.MORPH_ELLIPSE/` taille `(5,5),(3,3)`
L'élément structurant peut être construit manuellement en choisissant toute forme souhaitée.

Le gradient morphologique, le chapeau haut de forme, le chapeau bas de forme sont donnés respectivement par les fonctions :

```
cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel).
```

a. Transformations morphologiques sur des images binaires

- Charger l'image **blobs.png** présentée à la figure 1.

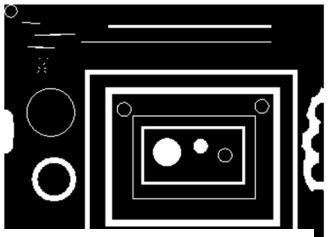


Figure 1 : image blobs

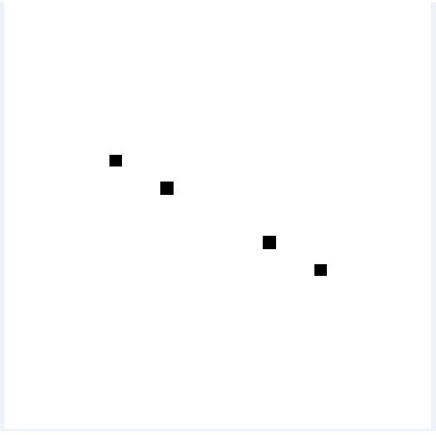
- Définir un élément structurant de forme rectangle et de taille 3.
- Appliquer et afficher une érosion, une dilatation, une fermeture et une ouverture tout en interprétant à chaque fois le résultat.
- Changer la taille et la forme de l'élément structurant à ligne horizontale de taille 5, puis à ligne verticale de taille 5, refaire les mêmes transformations précédentes. Interpréter.
- Pour éliminer les structures fines de cette image, quelle opération il faut appliquer et quel élément structurant est le mieux adapté (forme et taille).

b. Transformations morphologiques sur des images en niveaux de gris

- Charger l'image **liftingbody.png**.
- Appliquer et afficher les transformations morphologiques suivantes : Ouverture, Fermeture, Chapeau haut de forme (CHF), Chapeau bas de forme (CBF), à l'aide de deux éléments structurants différents. Le premier carré de taille 5, le deuxième une ligne horizontale de taille 5. Interpréter les deux résultats et comparer en vous aidant de profile_line.
- Extraire et Afficher le contour par la méthode du gradient morphologique (complet, interne et externe). Choisissez l'élément structurant convenable pour avoir un contour fin. Justifier.

II. Filtrage dans le domaine spectral

Dans un premier temps charger les deux images `lenabruitee.png` et `masqueLena.tif`, puis exécuter le script suivant qui permet de débruiter l'image dans le domaine fréquentiel.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
path = r'lenabruitee.png'
img = cv.imread(path,0)

pathmasq= r'masqueLena.tif'
maskI=cv.imread(pathmasq,0)
# apply DFT
dft = cv.dft(np.float32(img),flags = cv.DFT_COMPLEX_OUTPUT)

dft_shift = np.fft.fftshift(dft)
magnitude_spectrum =
20*np.log(cv.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
rows, cols = img.shape

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[:, :,0] = maskI[:,:]
mask[:, :,1] = maskI[:,:]

# apply mask and inverse DFT
fshift = dft_shift*mask

f_ishift = np.fft.ifftshift(fshift)
img_back = cv.idft(f_ishift)
img_back = cv.magnitude(img_back[:,:,0],img_back[:,:,1])
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('filtered image'), plt.xticks([]), plt.yticks([])
plt.show()

cv.waitKey(0)
# closing all open windows
cv.destroyAllWindows()
```

- Dans un deuxième temps, charger et afficher l'image **liftingbodybruite.png**. Calculer et Visualiser le module du spectre de Fourier. Interpréter.
- Créer un filtre adéquat pour éliminer le bruit de l'image.
- Réaliser le filtrage dans le domaine spectral (fréquentiel) et visualiser les images résultantes dans le domaine spatial.