Exercices
Semaine 15

Cours Turing

Exercice 1

Implémenter les 4 fonctions suivantes :

- `correlation(A, B)` : retourne la corrélation entre les tableaux A et B
- `convolution(A, B)` : retourne la convolution entre les tableaux A et B
- `convolution_over_2dimage(image, filtre)` : Applique, sur chaque partie de l'image, la convolution avec le filtre et retourne l'image résultante
- `convolution_over_3dimage(image, filtre)` : Applique, sur chaque partie de l'image, et pour chaque couleur, la convolution avec le filtre et retourne l'image résultante

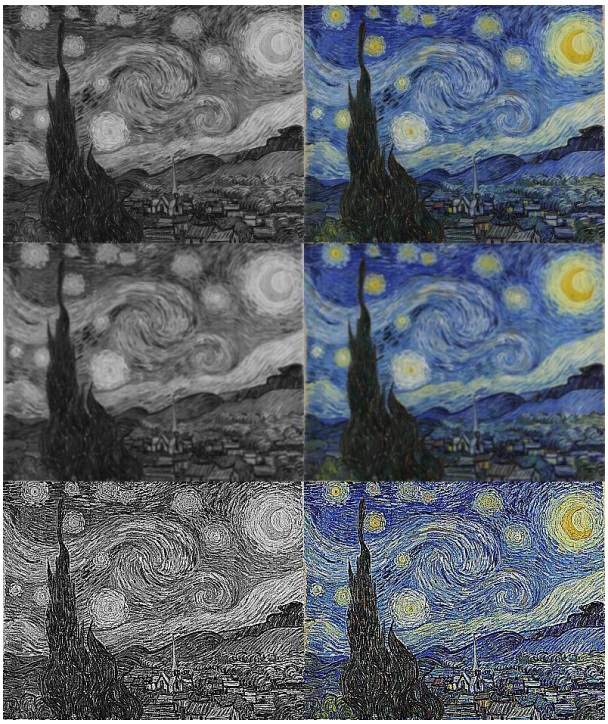
A noter que deux filtres sont déjà définis dans le code et sont donnés aux fonctions `convolution_over_2dimage` et `convolution_over_3dimage` afin de tester votre implémentation.

Indice

1. La fonction `np.rot90` permet de faire une rotation de 90 degrés sur un tableau
2. La fonction `np.sum` permet de faire la somme des éléments d'un tableau

Si les fonctions `correration` et `convolution` sont correctement implémentées, alors le terminal devra également afficher 285 et 165 comme pour les exemples montrés dans la théorie.

Résultat attendu



Mini-projet : Détection de contours

La détection de contours est le support de nombreux algorithmes de vision par ordinateur.

Il n'existe pas de méthode universelle pour détecter les contours d'une image, mais la méthode la plus utilisée est celle de Canny.

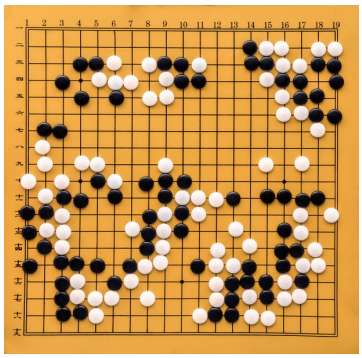


Figure 1: Image originale

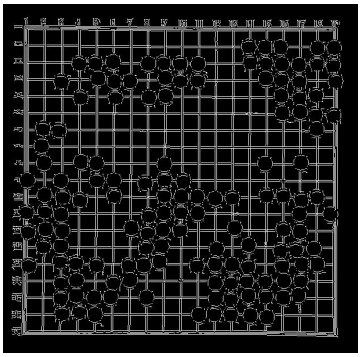


Figure 2: Detection de contours en utilisant la méthode de Canny

Dans ce mini-projet, vous allez implémenter la méthode de Canny. Afin de détecter les contours dans une image, la méthode de Canny consiste à suivre les étapes suivantes :

1. Transformer l'image en gris
2. Flouter l'image avec, par exemple, un filtre gaussien
3. Calculer la magnitude et le gradient de l'image à l'aide des filtres de Sobel
4. Supprimer les pixels qui ne sont pas maximaux
5. Appliquer un seuil sur l'image

Si vous trouvez que votre implémentation de la convolution est trop lente, vous pouvez utiliser la fonction `scipy.signal.convolve2d` qui est une implémentation optimisée de la convolution.

Il faudra installer la librairie `scipy` pour pouvoir utiliser la fonction `scipy.signal.convolve2d` :

```
pip3 install scipy
```

Et ajouter la ligne suivante au début de votre script :

```
from scipy.signal import convolve2d
```

Et l'utiliser comme ceci :

```
res = convolve2d(numpy_image, filtre, mode='valid')
```

Transformer l'image en gris

Vous pouvez utiliser la fonction `ntsc_grayscale` que vous avez implémentée la semaine dernière.

Alternativement, vous pouvez utiliser `.convert('L')` sur une image chargée avec la librairie PIL pour la convertir en gris.

Flouter l'image

Il faudra utiliser un filtre gaussien pour flouter l'image en utilisant la fonction de convolution implémentée dans l'exercice 1.

Le filtre gaussien G 3x3 est le suivant :

$$G = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Calculer la magnitude et le gradient de l'image à l'aide des filtres de Sobel

Les filtres de Sobel analysent l'image dans deux directions : horizontale et verticale.
Les filtres de Sobel sont les suivants :

Sobel_X =

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel_Y =

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Soit X, le résultat de la convolution entre l'image et le filtre Sobel_X et Y, le résultat de la convolution entre l'image et le filtre Sobel_Y. La magnitude et le gradient de l'image, selon Sobel, sont définis par :

$$Magnitude = \sqrt{X^2 + Y^2}$$

$$Gradient = \arctan 2 \frac{Y}{X}$$

La librairie `numpy` fournit les fonctions `numpy.sqrt` et `numpy.arctan2` pour calculer la racine carrée et l'arctangente de chaque élément d'un tableau numpy.

Supprimer les pixels qui ne sont pas maximaux

Grâce aux filtres de Sobel, nous avons pu calculer la magnitude et le gradient de l'image pour chaque pixel.

Pour chaque pixel, nous allons comparer sa magnitude avec celle de ses deux voisins dans la direction du gradient. Si la magnitude du pixel est plus grande que celle de ses voisins, alors nous gardons le pixel, sinon nous mettons sa magnitude à 0.

1 ↗	3 ↗	5 ↑	4 ↑	3 ↑	1 ↑
2 ↗	4 ↑	6 ↑	5 ↑	5 ↑	5 ↑
3 ↑	6 ↑	7 ↑	7 ↑	6 ↑	4 ↑
4 ↑	5 ↑	6 ↑	6 ↑	4 ↑	3 ↗
2 ↑	4 ↑	4 ↑	5 ↑	3 ↗	2 ↗

Figure 3: La valeur représente la magnitude du pixel et la flèche représente la direction du gradient.

Dans la figure 3 les pixels en bleu seront conservés car ils sont plus grands que leurs voisins dans la direction du gradient. Les pixels en blanc seront supprimés parce qu'ils sont plus petits que leurs voisins dans la direction du gradient.

La fonction `numpy.atan2` retourne des valeurs comprises entre $-\pi$ et π.

Soit G, le tableau des gradients et M, le tableau des magnitudes et (X, Y) les coordonnées du pixel courant alors :
$G_{[x,y]} \in [-\pi, \frac{-7\pi}{8}]$ ou $G_{[x,y]} \in [-\frac{\pi}{8}, \frac{\pi}{8}]$ ou $G_{[x,y]} \in [\frac{7\pi}{8}, \pi]$, alors on compare $M_{[x,y]}$ avec $M_{[x,y-1]}$ et $M_{[x,y+1]}$
$G_{[x,y]} \in [\frac{-7\pi}{8}, \frac{-5\pi}{8}]$ ou $G_{[x,y]} \in [\frac{\pi}{8}, \frac{3\pi}{8}]$, alors on compare $M_{[x,y]}$ avec $M_{[x-1,y+1]}$ et $M_{[x+1,y-1]}$
$G_{[x,y]} \in [\frac{-5\pi}{8}, \frac{-3\pi}{8}]$ ou $G_{[x,y]} \in [\frac{3\pi}{8}, \frac{5\pi}{8}]$, alors on compare $M_{[x,y]}$ avec $M_{[x-1,y]}$ et $M_{[x+1,y]}$
$G_{[x,y]} \in [\frac{-3\pi}{8}, \frac{-1\pi}{8}]$ ou $G_{[x,y]} \in [\frac{5\pi}{8}, \frac{7\pi}{8}]$, alors on compare $M_{[x,y]}$ avec $M_{[x-1,y-1]}$ et $M_{[x+1,y+1]}$

Vous pouvez utiliser `numpy.pi` pour obtenir la valeur de π.

Il est possible de convertir ces angles en degrés en multipliant par $\frac{180}{\pi}$ travailler avec l'intervalle $[-180°, 180°]$ si cela vous convient mieux.

- $\frac{\pm\pi}{8} = \pm 22.5°$
- $\frac{\pm 3 \cdot \pi}{8} = \pm 67.5°$
- $\frac{\pm 5 \cdot \pi}{8} = \pm 122.5°$
- $\frac{\pm 7\pi}{8} = \pm 157.5°$
- $\pm\pi = \pm 180°$

Appliquer un seuil sur l'image

Actuellement, l'image est composée de valeurs dont l'intensité varie grandement. Le but de cette étape est de simplifier cela, nous ne voulons utiliser que deux valeurs : 0 et 255.
0 signifie que le pixel ne fait pas partie du contour et 255 signifie que le pixel fait partie du contour.

La fonction `threshold` prend en paramètres un tableau numpy et un seuil, vous pouvez utiliser un seuil de, par exemple de 200, mais vous êtes encouragés à en essayer plusieurs afin de voir l'impact de ce paramètre

Pour chaque pixel du tableau, si la valeur du pixel est plus grande que le seuil, alors la valeur du pixel est mise à 255, sinon elle est mise à 0.

Résultats attendus

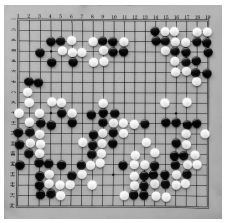


Figure 4: Gris

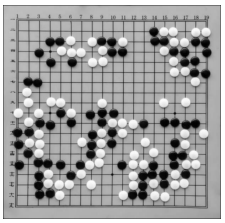


Figure 5: Flou

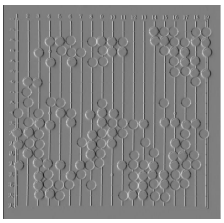


Figure 6: Sobel x

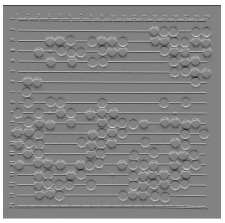


Figure 7: Sobel y

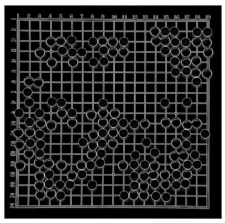


Figure 8: Magnitude

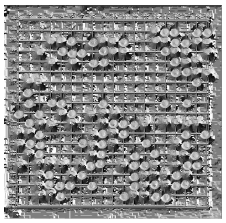


Figure 9: Gradient

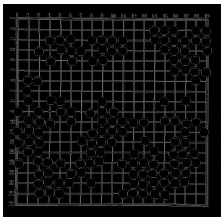


Figure 10: Suppression des pixels non maximaux

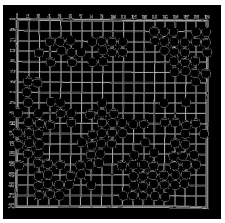


Figure 11: Threshold