TD-TP 2

Objectifs :

- étude du lissage linéaire et non-linéaire

- étude du gradient avec l'opérateur Sobel

Partie TD

Lissage par moyenne et médiane

On donne l'image ci-contre:

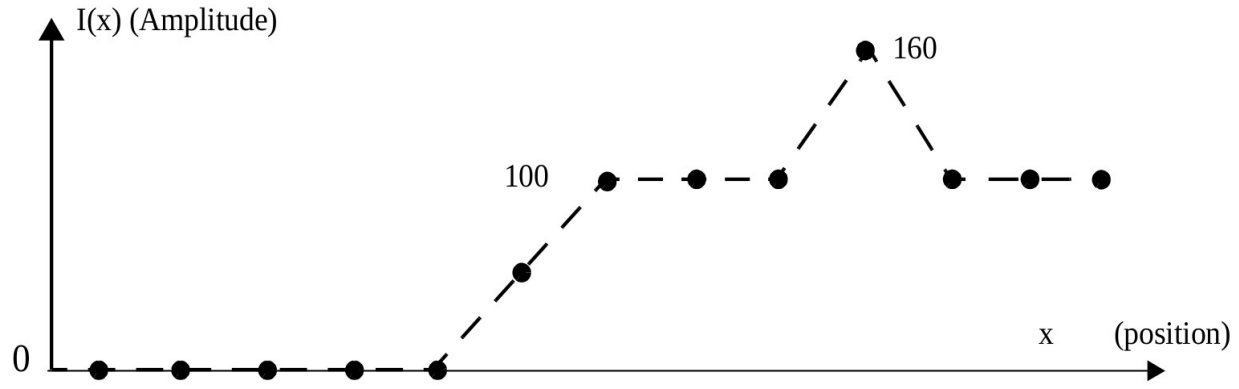
10	11	12	10	12
11	**12**	**12**	**11**	11
11	**11**	**12**	**13**	11
10	**12**	**10**	**13**	12
10	11	12	13	12

1 - Donner le résultat d'un filtrage moyenneur 3x3 sur cette image sur les neufs pixels du centre de l'image. On arrondira le résultat au plus proche entier.

2 - Comment est modifié le résultat si l'intensité du pixel en bas à droite passe de 12 à 192.

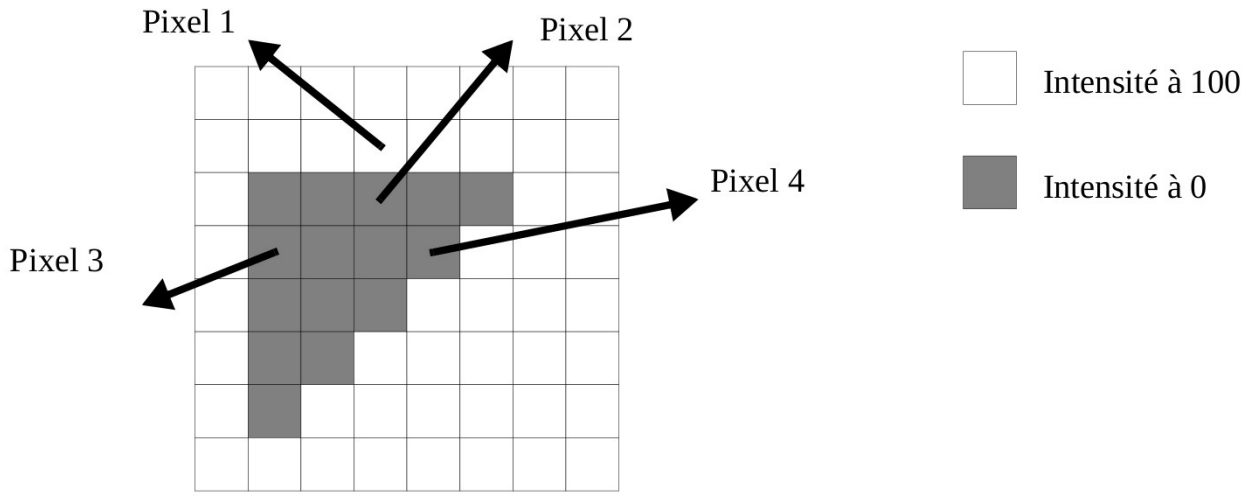
3 - Reprendre les questions 1 et 2 en utilisant un filtre médian 3x3 à la place du filtre moyenneur.

4 - Comparer le résultat du filtrage médian et du filtrage moyenneur sur le signal suivant que l'on peut voir comme un contour vertical (on prendra un masque de taille 1x3).



Gradient par opérateur Sobel

5 - On donne l'image ci-dessous :



Calculer le résultat de l'application de l'opérateur de Sobel sur les 4 pixels 1, 2, 3 et 4 indiqués sur la figure. Commenter le résultat.

6 - Pour ces 4 pixels, calculer le module et la direction du gradient.

7 - On propose un calcul du gradient à partir des masques ci-dessous. Comment définir simplement le module et la direction du gradient ?

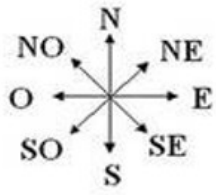
Exemple : Robinson (3)

1	1	0
1	0	-1
0	-1	-1

1	1	1
0	0	0
-1	-1	-1

0	1	1
-1	0	1
-1	-1	0

1	0	-1
1	0	-1
1	0	-1



-1	0	1
-1	0	1
-1	0	1

0	-1	-1
1	0	-1
1	1	0

-1	-1	-1
0	0	0
1	1	1

-1	-1	0
-1	0	1
0	1	1

Partie TP en Python

Lissage par moyenne et médiane

1 - Lire la documentation en ligne sur la fonction matlab cv2.filter2D (documentation OpenCV en ligne pour les versions 3.x,

exemple : https://docs.opencv.org/3.0.0/) . Tester cette fonction pour filtrer une image avec un moyenneur 3x3, puis 5x5. On pourra ajouter un bruit à l'image initiale avec les générateurs de bruit proposés dans la bibliothèque numpy.random puis tester le débruitage avec les fonction cv2.filter2D et cv2.GaussianBlur. Exemple d'utilisation :

blurred_img = cv2.GaussianBlur(img_in,(5,5),0, 0.02)

crée l'image ***blurred_img*** en ajoutant à l'image ***img_in*** un bruit gaussien de moyenne nulle et de variance 0.02.

2 - Simulez ensuite un défaut de capteur typique qui soit éteint un pixel, soit le sature. On peut alors parler de bruit 'Sel et Poivre'. Pour cela, générez une matrice de bruit de taille similaire à celle de votre image. Pour cela, regardez la documentation numpy.random et choisissez le bon générateur de bruit. Codez ensuite une méthode permettant de forcer les pixels de bruit inférieurs à une borne min (exemple, 20% de la valeur maximale du bruit) et forcez/éteignez les pixels correspondants dans l'image à la valeur 0. De même, identifiez les pixels de la matrice de bruit supérieurs à un seuil max (exemple, 80% de la valeur maximale du bruit) et forcez/saturez les pixels correspondants de l'image à leur propre valeur maximale : 255. Appliquez ensuite le filtre moyenne.

3 - Tester le filtre médian avec différents bruits et différents niveaux de bruits (voir fonction opencv cv2.medianBlur).

4 - Comparer les résultats obtenus entre les deux filtres et les types de bruits:

1. visuellement. Trouvez vous une correspondance entre un type de bruit et un type de filtrage ?
2. de façon quantifiée: proposez une solution permettant de mesurer la différence entre l'image initiale non bruitée et l'image bruitée "corrigée par le filtrage". Peut on déduire une mesure de performance du filtrage correctif ?

Gradient par opérateur Sobel

3 - En utilisant la fonction cv2.filter2D et les masques de Sobel, calculer les composantes Gx et Gy du gradient. On utilisera l'image «coins_ChastanaCoin.jpg», puis avec l'image «coins_Dutch_17_18_C_coins_obv.jpg» et d'autres de votre choix.
ATTENTION 1 : les images résultats peuvent avoir des valeurs en dehors de la plage [0 - 255]. Pour la visualisation des résultats, il faut normaliser les valeurs entre 0 et 255. On utilisera donc la fonction de normalisation correspondante réalisée au TP1.

ATTENTION 2 : attention au type de vos images traitées par ces filtres... d'ailleurs, quel type doit être utilisé ?

Analyser les résultats obtenus.

4 - Calculer le module et la direction du gradient. Analyser les résultats obtenus.

5 - Lire l'aide sur la fonction cv2.Sobel, comparez vos résultats (valeurs, temps de calcul) avec ceux obtenus avec cette fonction. Analysez.

6 - Lire l'aide sur la fonction cv2.Canny. Utiliser cette fonction pour calculer les contours d'une image. De quelle nature est l'image résultante (distribution des valeurs). Peut on obtenir un résultat similaire avec Sobel ? Tentez une explication.

Pour ceux qui sont en avance :

7 - Etendre la détection des contours au cas des images couleur en utilisant une approche «marginale» suivie d'une fusion des contours obtenus sur chaque composante. Tester différents opérateurs de fusion.