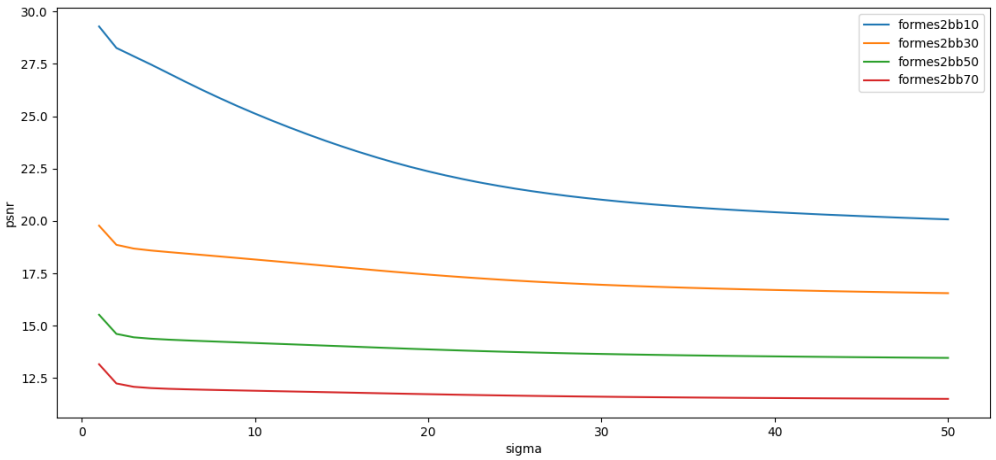
Rapport du TP1

EZ-ZEJJARI Imad
BENLHABIB Youssef

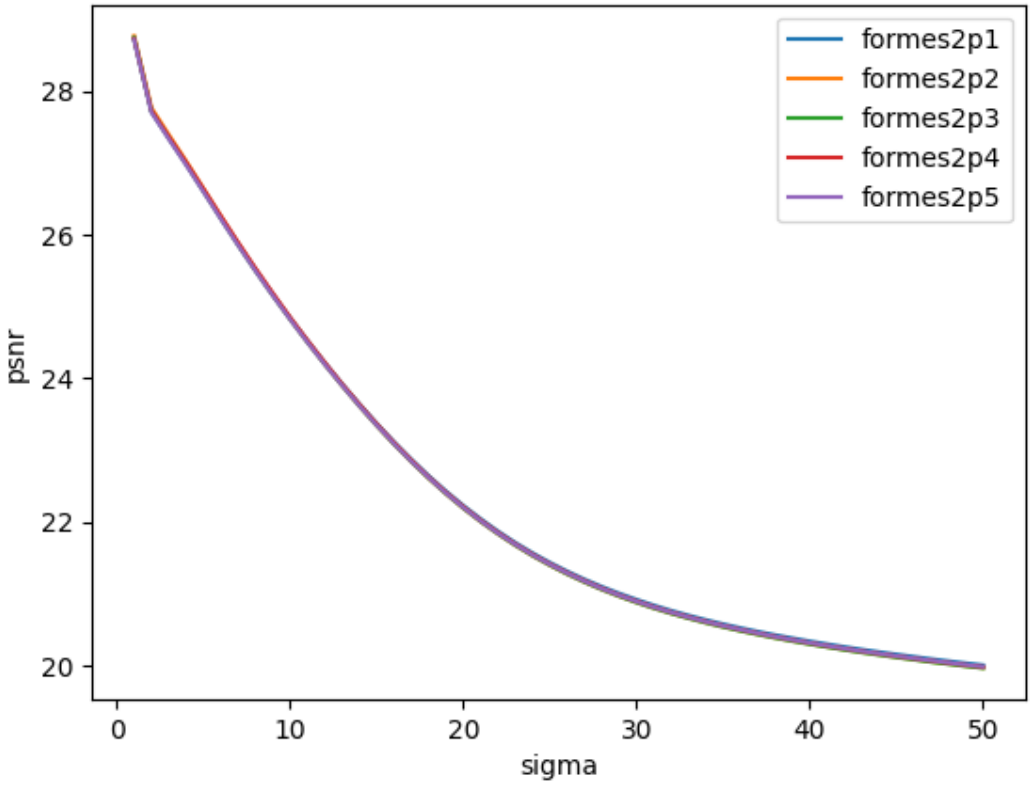
A. FFT, filtrage fréquentiel et convolution

1. Filtrage fréquentiel gaussien

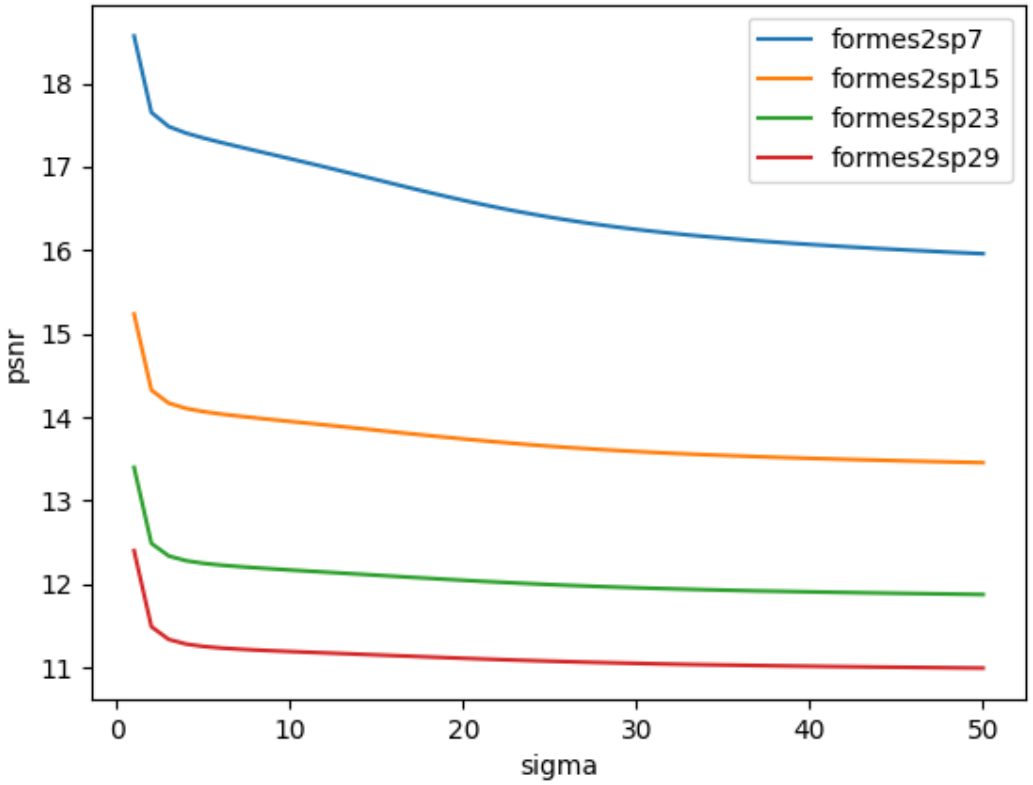
Le code est implémenté dans **gaussien.c** et le test dans **test_gaussien.c.**
On trace l'évolution du PSNR pour quelques images en variant le bruit et la valeur de sigma afin de tester l'efficacité du filtre.
Les courbes suivantes résument tous les résultats obtenus:



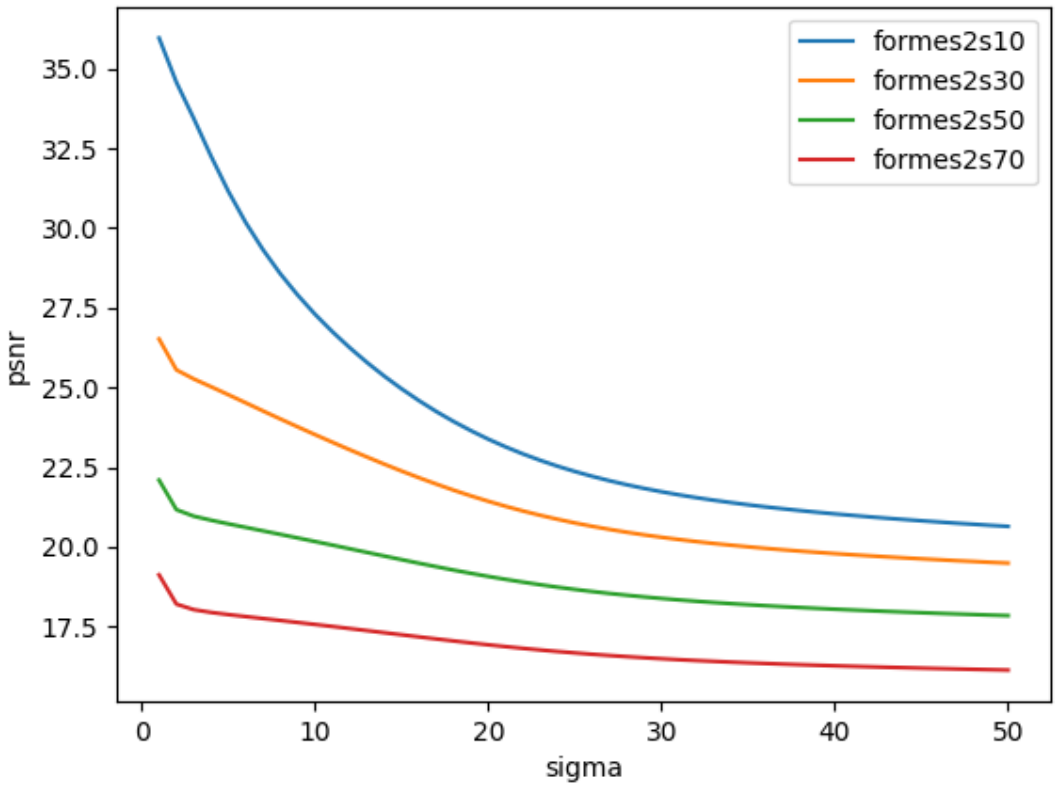
Gaussien.pgm


formes2p1
formes2p2
formes2p3
formes2p4
formes2p5
28
26
24
22
20
psnr
0
10
20
30
40
50
sigma

Poisson.pgm



Poivre_sel.pgm



Speeckle.pgm

- La qualité du lissage dépend de la valeur de sigma.
- Plus sigma est petit, plus on obtient une bonne qualité du lissage.
- Pour les images de type **poisson**, la qualité du lissage ne dépend pas de la quantité de bruit introduite.
- Par contre, pour les autres images, plus le bruit est important, plus le lissage est moins pertinent.
- Ainsi, le filtre gaussien est efficace pour le bruit **poissonien.**
- De plus, l'effet de sigma sur la qualité du lissage devient moins important si on introduit trop de bruit.

2. Filtrage spatiale gaussien

Le code est implémenté dans **gaussienspatial.c** et le test dans **test_gaussien_spatial.c.**
Dans ce filtre, on complète l'image comme indiqué dans l'énoncé par sa valeur dans le côté opposé, ici un exemple de sortie exécutée sur formes2bb5.pgm:

gaussspatialbb5result.pgm

Cette fonction a une complexité **O (n.m.N.M)** où:
n, m :taille du filtre
N, M: taille de l'image
Notre implémentation utilise un filtre non séparable, ce qui justifie sa complexité par rapport au filtrage gaussien, néanmoins elle s'avère intéressante dans certains cas expliqué lors de la comparaison entre les deux méthodes.

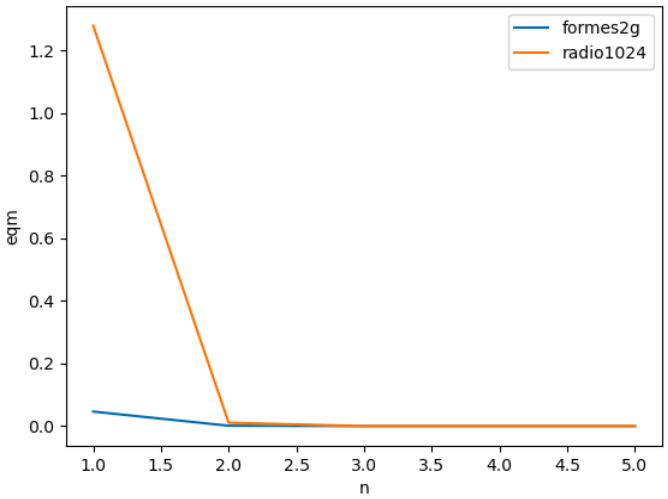
➔ **L'écart quadratique moyen:**

Les images sont dans le répertoire **courbes**, on a utilisé le script **eqm.py** pour le tracé, après avoir obtenu les valeurs depuis l'exécution de **valeureqm.c.**

Dans ces courbes, on a tracé l'allure de la fonction eqm en fonction de la taille du filtre en balayant l'intervalle 1 à 6*sigma.

- **Pour sigma = 1:**

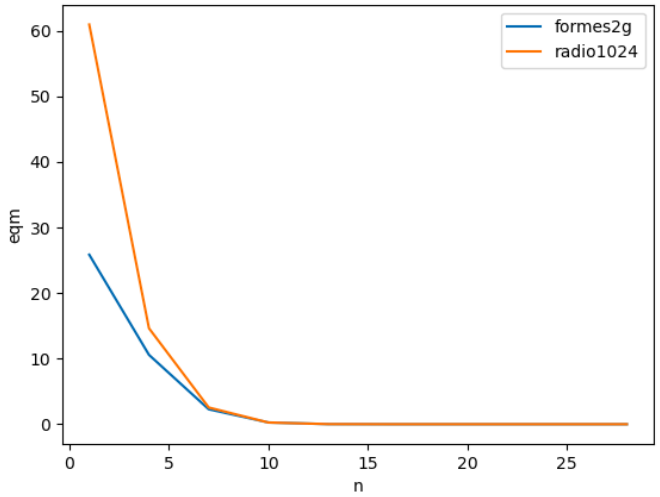
On a tracé l'erreur pour sigma égal à 1, mais puisque on utilise le paramètre n comme entier, on obtient que quelques valeurs ce qui rend la courbe moins lisse:



eqm_sigma1.png

On voit que pour la taille du masque égal à 4*sigma, l'erreur est nul pour les deux courbes, ce qui nous ramènent à donner l'hypothèse que 4* sigma sera la fonction minimale pour laquelle on aura une eqm nulle.

- **Pour sigma = 5:**



eqm_sigma5.pgm

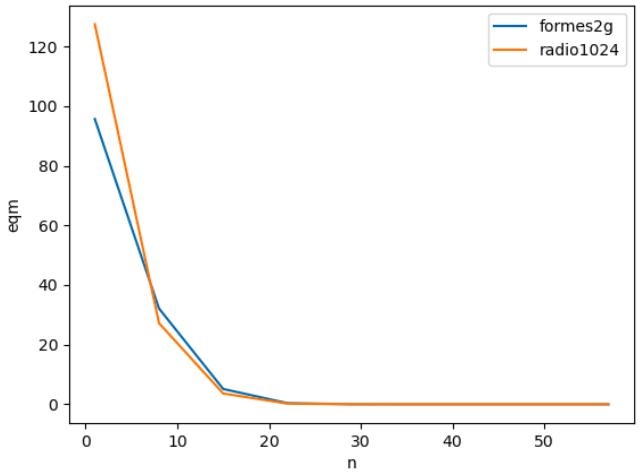
Valeur de eqm:

n = 19 : `0.000006 / 0.000004`

`n = 20: 0 / 0`

L'erreur quadratique est grande pour un filtre de petite taille surtout pour l'image de grande taille radio1024. Cependant, les deux courbes convergent rapidement vers 0, et deviennent égal à 0 exactement lorsque n = 4*sigma, ce qui permet de valider l'hypothèse faite au début.

- **Pour sigma = 10:**



eqm_sigma10.pgm

La remarque présentée avant pour sigma égal à 5 s'applique aussi à sigma égal 10, et on voit que eqm converge plus vite pour formes2g, mais la valeur de l'erreur devient nulle pour 40, ce qui valide notre hypothèse.

Donc on déduit que la fonction W convenable pour les deux images, et qui minimise au plus, l'erreur pour les deux images dans l'intervalle 1 à 6* sigma comme taille du filtre est : proche de la valeur 4 *sigma.

3. **Complexité et comparaison des 2 méthodes**

Pour une taille de masque égale à 4 et pour des petites valeurs de sigma, on compare les temps de calcul des deux implémentations pour des tailles d'images différentes.

- **Sigma = 1**:

	Gaussien	Gaussien spatial
formes2g.pgm	0.04	0.19
radio1024.pgm	1.12	2.29
radio1.pgm	0.18	0.5

- **Sigma = 0.5:**

	Gaussien	Gaussien spatial
formes2g.pgm	0.04	0.08
radio1024.pgm	1.11	0.95

radio1.pgm	0.25	0.19

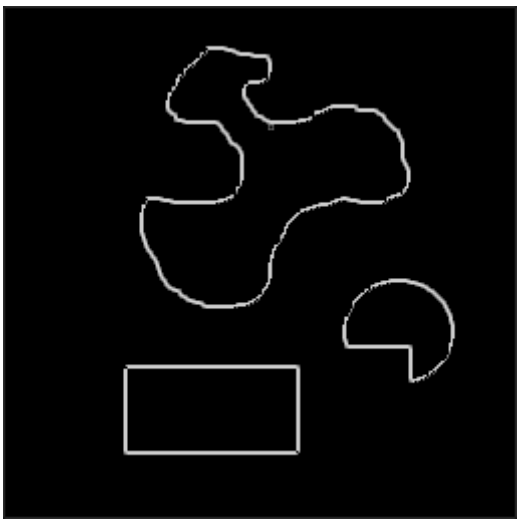
- On constate que pour des images de grande taille (1024*1024), il est préférable d'utiliser la version spatiale qui prend moins de temps au niveau de la complexité.
- Théoriquement, ceci est totalement prévu vu que la taille du masque **(n * m)** devient inférieure au logarithme de la taille de l'image **ln(N * M).**

B. Détection de contour

1. Par le gradient (Convolution spatiale Prewitt)

Le code est implémenté dans **contour1.c** et le test dans **test_contour1.c**
On effectue la détection du contour de l'image **formes2.pgm** par l'opérateur du premier ordre (gradient):

formes2.pgm



contour1.pgm

Ici, on détecte le contour par la méthode du premier ordre implémenté dans le fichier contour1.c, ou on calcule le gradient de l'image, et on voit si il est maximal suivant la direction du gradient, et il est mi à zero sinon, c'est pour cela on voit la couleur noir, et le contour est bien detecté en blanc. On utilise le filtre de Prewitt pour calculer le gradient.

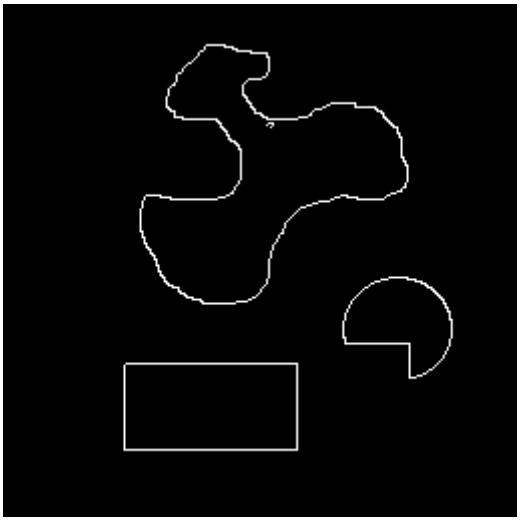
2. Par le laplacien (convolution spatiale):

Le code est implémenté dans **contour2.c** et le test dans **test_contour2.c**
De la même façon que l'extraction du contour par le gradient, on détecte le contour par des convolutions discrètes entre l'image et le masque suivant:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

En réalisant un lissage gaussien, puis en détectant les passages par zéro du Laplacien, on extrait le contour.

formes2.pgm

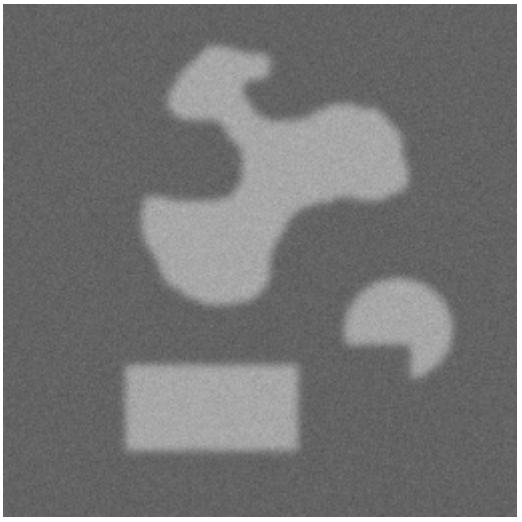


contour2.pgm

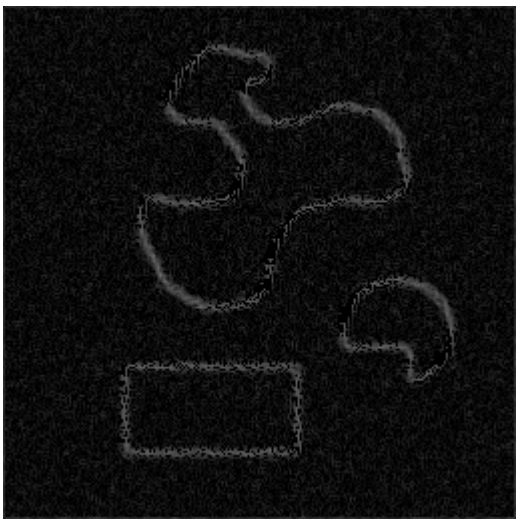
3. Comparaison:

- D’un point de vue qualitatif, la détection du contour par gradient et par laplacien donne des résultats identiques pour l’image sans bruit **formes2.pgm.**
- Par contre, quand l’image est bruitée, on obtient une extraction plus pertinente par la méthode du gradient.

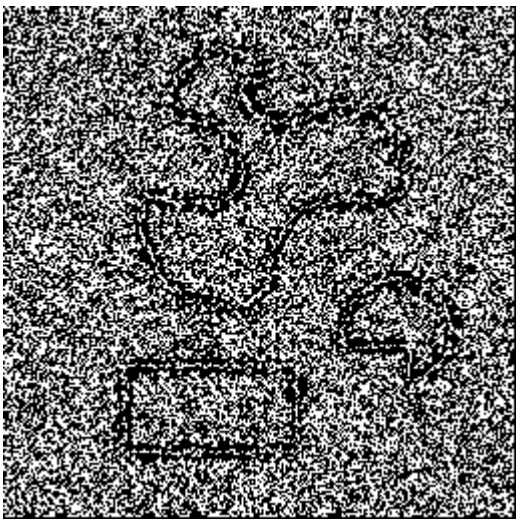
- On prend par exemple l'image de type gaussien **formes2bb5.pgm** et on extrait son contour à l'aide des deux méthodes:



formes2bb5.pgm



contour1bruit.pgm



contour2bruit.pgm

- Ainsi, la qualité de la détection par le laplacien diminue lorsqu'il y a du bruit. L'opérateur de deuxième ordre est sensible au bruit. Il est donc préférable de procéder par

une extraction avec la première méthode quand l’image est bruitée.