Une image contenant Police, Graphique, logo, graphisme

Description générée automatiquement

**Département de génie logiciel et des T.I.**

Rapport de Laboratoire

|  |  |
| --- | --- |
| **Numéro du laboratoire** | Laboratoire 2 |
| **Nom du laboratoire** | Filtrage |
| **Étudiant(s)** | Olivier, Granger- Hotte  Gildor Makesa Mvuemba |
| **Code(s) permanent(s)** | MAKM87260201  GRAO89120006 |
| **Numéro d’équipe** | 06 |
| **Cours** | GTI 411 |
| **Session** | Hiver 2025 |
| **Groupe** | S20251-GTI411-01 |
| **Chargé(e) de laboratoire** | Lucas Mercier |
| **Date** | 24 février 2025 |

[Introduction 2](#_Toc1148520543)

[Outils et concepts 3](#_Toc1991554487)

[Le principe du filtrage passe-haut et passe-bas 3](#_Toc161211507)

[Manipulation de l’image avec les filtrages haut et basse 4](#_Toc1042923563)

[Le filtrage dans le domaine fréquentiel 4](#_Toc1599459256)

[Implémentation 5](#_Toc1616217367)

[Algorithmes utilisés 5](#_Toc657604514)

[Filtre spatial 5](#_Toc1370813361)

[Filtre de Canny 7](#_Toc892847532)

[Filtre de fréquence 8](#_Toc251152899)

[Librairies utilisées 10](#_Toc628610643)

[Résultats et discussion 10](#_Toc1208351317)

[Résultats 11](#_Toc370915642)

[Spatial Filters 11](#_Toc2134042240)

[Canny algorithm 14](#_Toc1823212462)

[Frequency Filters 14](#_Toc1070630596)

[Difficultés rencontrées 17](#_Toc161435775)

[Améliorations potentielles 17](#_Toc1690347889)

[Conclusion 18](#_Toc800238267)

[Annexe : Manuel d’utilisateur 18](#_Toc1106638260)

[Exécution du code 18](#_Toc601612875)

# Introduction

L’utilisation de filtres en traitement d’images est essentielle dans de nombreux domaines allant de la vision par ordinateur à l’imagerie médicale. OpenCV, une bibliothèque open-source largement utilisée, permet d’appliquer différents types de filtres pour améliorer ou extraire des caractéristiques d’une image.Le filtrage consiste à modifier les valeurs des pixels d'une image afin d'améliorer son aspect ou d'en extraire des caractéristiques. En pratique, il s'agit de créer une nouvelle image en se servant des valeurs des pixels de l'image d'origine. Par conséquent, ce laboratoire vise à comprendre et implémenter divers filtres dans les domaines spatial et fréquentiel afin d’explorer leurs effets sur des images numériques.

Plus précisément, l’objectif de ce laboratoire est d’acquérir une compréhension pratique des filtres d’image en les mettant en œuvre avec OpenCV. Pour cela, plusieurs défis doivent être relevés : manipuler l’interface graphique existante afin d’appliquer correctement les filtres spatiaux (gaussien, Sobel, médian) tout en gérant les bordures et la conversion des valeurs filtrées ; implémenter le filtre de Canny en ajustant ses paramètres, notamment la taille du filtre gaussien et les seuils, afin d’obtenir des contours précis ; et appliquer des filtres fréquentiels en transformant les images dans le domaine de Fourier, puis en analysant les différences entre les filtres passe-bas et passe-haut, notamment avec les profils idéaux et Butterworth. Ces fonctionnalités sont largement utilisées dans de nombreuses applications pratiques. Elles sont utilisées dans le domaine de la vision par ordinateur pour l’amélioration et l’analyse d’images ou même en imagerie médicale pour le traitement des scanneurs et IRM.

Ce rapport est structuré en plusieurs sections afin d’explorer en détail les différentes étapes du traitement d’images par filtrage. Nous débuterons par une présentation des outils et concepts fondamentaux liés aux filtres spatiaux et fréquentiels. Ensuite, nous détaillerons l’implémentation des filtres spatiaux, notamment les filtres gaussiens, Sobel et médian, ainsi que l’algorithme de Canny. La section suivante portera sur l’application des filtres fréquentiels, en mettant l’accent sur les transformations de Fourier et les filtres passe-bas et passe-haut, y compris les filtres idéaux et Butterworth. Nous analyserons ensuite les résultats obtenus et les défis rencontrés lors de l’implémentation. Enfin, nous conclurons en résumant les apprentissages clés et en proposant des améliorations potentielles.

# Outils et concepts

Le filtrage en traitement d'images consiste à ajuster les valeurs des pixels d'une image dans le but d'améliorer sa qualité, d'enlever le bruit ou de préserver des caractéristiques importantes, comme les contours. Les filtres passe-haut et passe-bas sont des concepts fondamentaux dans ce domaine.

## Le principe du filtrage passe-haut et passe-bas

Le filtrage passe-haut et le filtrage passe-bas sont des techniques qui manipulent les différentes fréquences présentes dans une image

1. **Filtrage Passe-haut** : Ce type de filtre laisse passer les hautes fréquences, c'est-à-dire les variations rapides d'intensité, comme les détails fins et les bords nets. Il atténue les basses fréquences, qui correspondent aux variations lentes (par exemple, les grandes structures ou les zones homogènes). Le filtre passe-haut est ainsi particulièrement utilisé pour mettre en évidence les contours et les détails fins d'une image.
2. **Filtrage Passe-bas :** À l'inverse, le filtrage passe-bas laisse passer les basses fréquences (variations lentes, grandes structures) et atténue les hautes fréquences (détails fins, bruits). Il permet de conserver les variations lentes (zones uniformes, grandes structures), tout en supprimant les détails fins et les bruits. Ce type de filtrage est souvent utilisé pour flouter ou lisser une image.

## Manipulation de l’image avec les filtrages haut et basse

1. **Passe-haut** : Le contraste est souvent plus accentué, ce qui peut donner une sensation de **détails plus nets et plus sombres** sur certaines parties de l’image. L'image devient plus "détaillée", mais avec moins de douceur. Les bords deviennent plus nets et les détails (comme les textures) sont accentués. Le filtre "supprime" les zones uniformes et lisses.
2. **Passe-bas** : L'image devient plus **uniforme et lisse**, avec un potentiel effet de **flou**, mais elle peut aussi paraître **moins brillante ou plus "adoucie"**. L'image devient plus "lisse" et douce. Les détails fins, les bruits ou les petites variations sont atténués, et l’image semble avoir un **flou** appliqué, tout en préservant les grandes structures.

## Le filtrage dans le domaine fréquentiel

Le filtrage dans le domaine fréquentiel se base sur la transformée de Fourier, qui permet de décomposer une image en différentes composantes fréquentielles (hautes et basses fréquences). Cette approche est particulièrement efficace pour appliquer les filtres passe-haut et passe-bas de manière précise :

1. **Filtrage Passe-haut dans le domaine fréquentiel** : Lorsqu’une image est transformée dans le domaine fréquentiel, elle est représentée sous forme de fréquences. Le filtrage passe-haut consiste à supprimer ou réduire les basses fréquences (représentant les variations lentes dans l’image), et à conserver les hautes fréquences (qui correspondent aux détails fins, textures et bords). En d'autres termes, ce filtre laisse passer les éléments qui changent rapidement dans l'image, tout en éliminant les parties plus stables ou uniformes.
2. **Filtrage Passe-bas dans le domaine fréquentiel** : Le filtrage passe-bas, à l'inverse, consiste à éliminer ou atténuer les hautes fréquences, qui correspondent aux détails fins et au bruit. Cela permet de conserver les basses fréquences, représentant les grandes structures et les transitions douces dans l'image. En appliquant ce filtre dans le domaine fréquentiel, l’image subit un lissage qui réduit les variations rapides de pixel, créant ainsi un effet de flou ou de douceur.

# Implémentation

## Algorithmes utilisés

### Filtre spatial

Cette section décrit surtout les différents types de filtres Sobel utilisés comme leur implémentation est plus intéressante.

#### Filtre Sobel-Y

SI kernel\_size == 0 ALORS

update\_ksize = 0

SINON

update\_ksize = max(0, kernel\_size | 1)

FIN SI

Définir border\_methods avec différentes méthodes de gestion des bords

grad\_y = SOBEL(image, dx=0, dy=1, ksize=update\_ksize, borderType=update\_border\_type)

SELON range\_method FAIRE

CAS "Abs and normalize to 255"

sobel\_image = ABS(grad\_y)

sobel\_image = NORMALIZE(sobel\_image, 0, 255, NORM\_MINMAX)

CAS "Abs and normalize 0 to 255"

sobel\_image = ABS(grad\_y)

sobel\_image = CLIP(sobel\_image, 0, 255)

CAS "Normalize 0 to 255"

sobel\_image = NORMALIZE(grad\_y, 0, 255)

CAS "Clamp 0 ... 255"

sobel\_image = CLIP(grad\_y, 0, 255)

FIN SELON

Retourner sobel\_image

#### Filtre Sobel-X

SI kernel\_size == 0 ALORS

update\_ksize = 0

SINON

update\_ksize = max(0, kernel\_size | 1)

FIN SI

Définir border\_methods avec différentes méthodes de gestion des bords

grad\_x = SOBEL(image, dx=1, dy=0, ksize=update\_ksize, borderType=update\_border\_type)

SELON range\_method FAIRE

CAS "Abs and normalize to 255"

sobel\_image = ABS(grad\_x)

sobel\_image = NORMALIZE(sobel\_image, 0, 255, NORM\_MINMAX)

CAS "Abs and normalize 0 to 255"

sobel\_image = ABS(grad\_x)

sobel\_image = CLIP(sobel\_image, 0, 255)

CAS "Normalize 0 to 255"

sobel\_image = NORMALIZE(grad\_x, 0, 255)

CAS "Clamp 0 ... 255"

sobel\_image = CLIP(grad\_x, 0, 255)

FIN SELON

Retourner sobel\_image

#### Filtre Sobel complet

SI kernel\_size == 0 ALORS

update\_ksize = 0

SINON

update\_ksize = max(0, kernel\_size | 1)

FIN SI

Définir border\_methods avec différentes méthodes de gestion des bords

grad\_x = SOBEL(image, dx=1, dy=0, ksize=update\_ksize, borderType=update\_border\_type)

grad\_y = SOBEL(image, dx=0, dy=1, ksize=update\_ksize, borderType=update\_border\_type)

magnitude\_sobel\_image = MAGNITUDE(grad\_x, grad\_y)

SELON range\_method FAIRE

CAS "Abs and normalize to 255"

sobel\_image = CONVERT\_SCALE\_ABS(magnitude\_sobel\_image)

CAS "Abs and normalize 0 to 255"

sobel\_image = NORMALIZE(ABS(magnitude\_sobel\_image), 0, 255, NORM\_MINMAX)

CAS "Normalize 0 to 255"

sobel\_image = NORMALIZE(magnitude\_sobel\_image, 0, 255, NORM\_MINMAX)

AUTRE

sobel\_image = CLIP(magnitude\_sobel\_image, 0, 255)

FIN SELON

Retourner sobel\_image

### Filtre de Canny

Une des fonctions intéressantes du filtre de Canny à décortiquer est celle des maximums locaux. Les autres fonctions de cette classe utilisent plutôt la librairie d'OpenCV afin de faire les changements nécessaires.

#### Fonction des maximums locaux

SI l'image a 3 canaux ALORS Convertir en niveaux de gris

Définir pad\_taille comme la moitié de neighborhood\_size

Créer une image avec les bords réfléchis

Créer une image locmax de même taille que l'image d'entrée

POUR chaque pixel (x, y) de l'image

Extraire la région de taille neighborhood\_size autour du pixel

SI la valeur du pixel est le maximum de la région ET qu'elle n'est pas le minimum ALORS

Marquer le pixel comme maximum local (255) FIN SI FIN POUR

Convertir locmax en RGB

Retourner locmax\_rgb

### Filtre de fréquence

Cette section décrit les algorithmes utilisés dans la partie 3 de ce laboratoire.

#### Filtre passe-bas de Butterworth

Obtenir dimensions de l'image en (rows, cols)

Définir centre\_x = rows // 2

Définir centre\_y = cols // 2

POUR chaque pixel (x, y)

distance = √((x - centre\_x)² + (y - centre\_y)²)

filtre(x, y) = 1 / (1 + (distance / cutoff\_freq)^(2 \* n))

FIN POUR

Retourner filtre

#### Filtre passe-haut de Butterworth

Obtenir dimensions de l'image en (rows, cols)

Définir centre\_x = rows // 2

Définir centre\_y = cols // 2

POUR chaque pixel (x, y)

distance = √((x - centre\_x)² + (y - centre\_y)²)

filtre(x, y) = 1 / (1 + (cutoff\_freq / MAX(distance, ϵ))^(2 \* n))

FIN POUR

Retourner filtre

#### Filtre passe-bas idéal

Obtenir dimensions de l'image en (rows, cols)

Définir centre\_x = rows // 2

Définir centre\_y = cols // 2

POUR chaque pixel (x, y)

distance = √((x - centre\_x)² + (y - centre\_y)²)

filtre(x, y) = SI distance ≤ cutoff\_freq ALORS 1 SINON 0

FIN POUR

Retourner filtre

#### Filtre passe-haut idéal

Retourner (1 - ideal\_lowpass\_filter(image, cutoff\_freq))

## Librairies utilisées

Nous avons surtout utilisé la librairie OpenCV pour ce laboratoire.

Cette librairie nous a permis de lire les images depuis des chemins de fichiers, puis de les traduire en matrice de couleurs afin de faire les manipulations nécessaires.

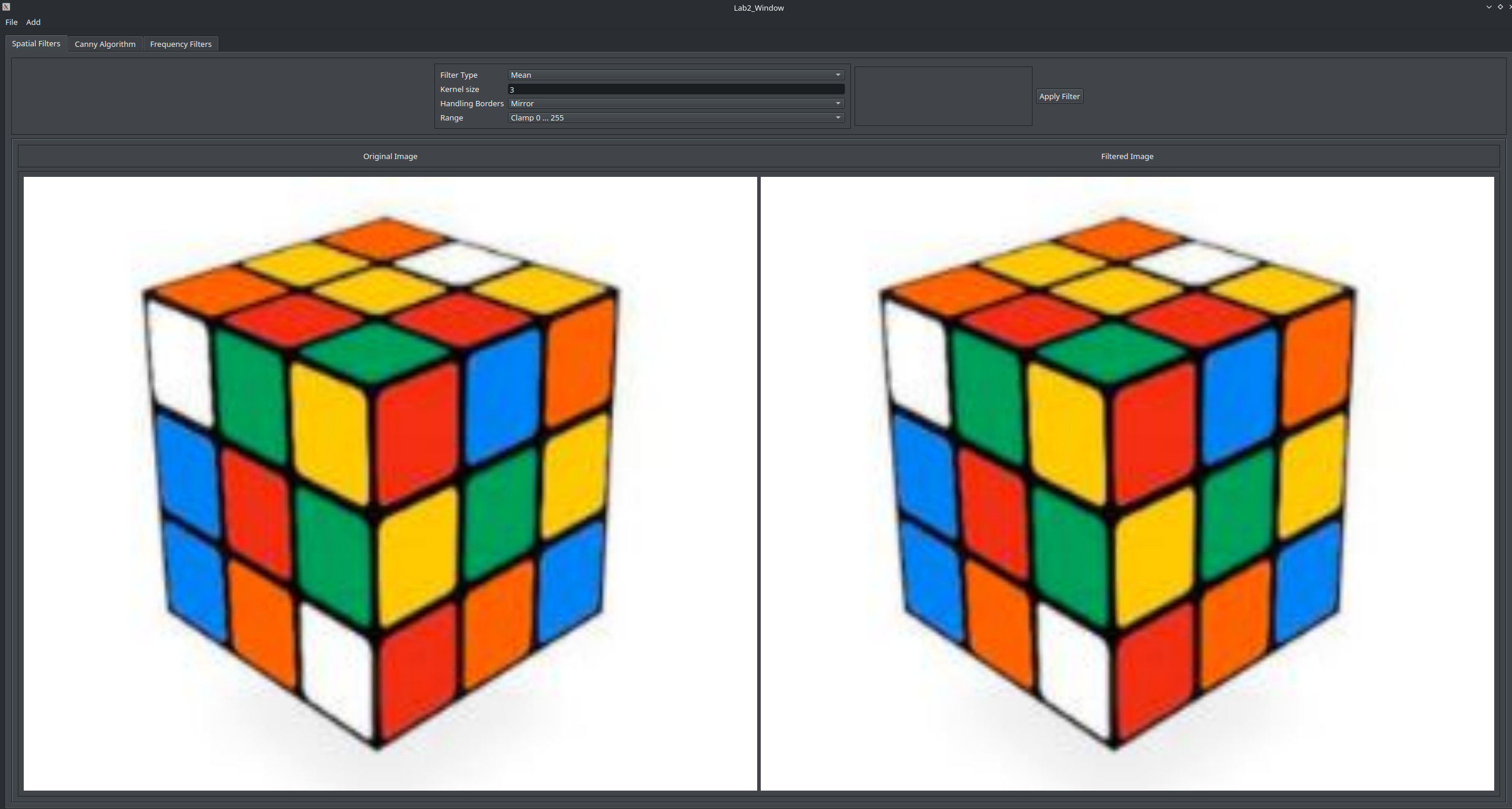
Nous avons aussi pu appliquer certains filtres de manière plus automatique grâce aux fonctions préfaites par cette librairie, ce qui nous a permis d’écrire du code clair et concis, tout en profitant pleinement de la puissance de ces algorithmes et de leur configuration.

# Résultats et discussion

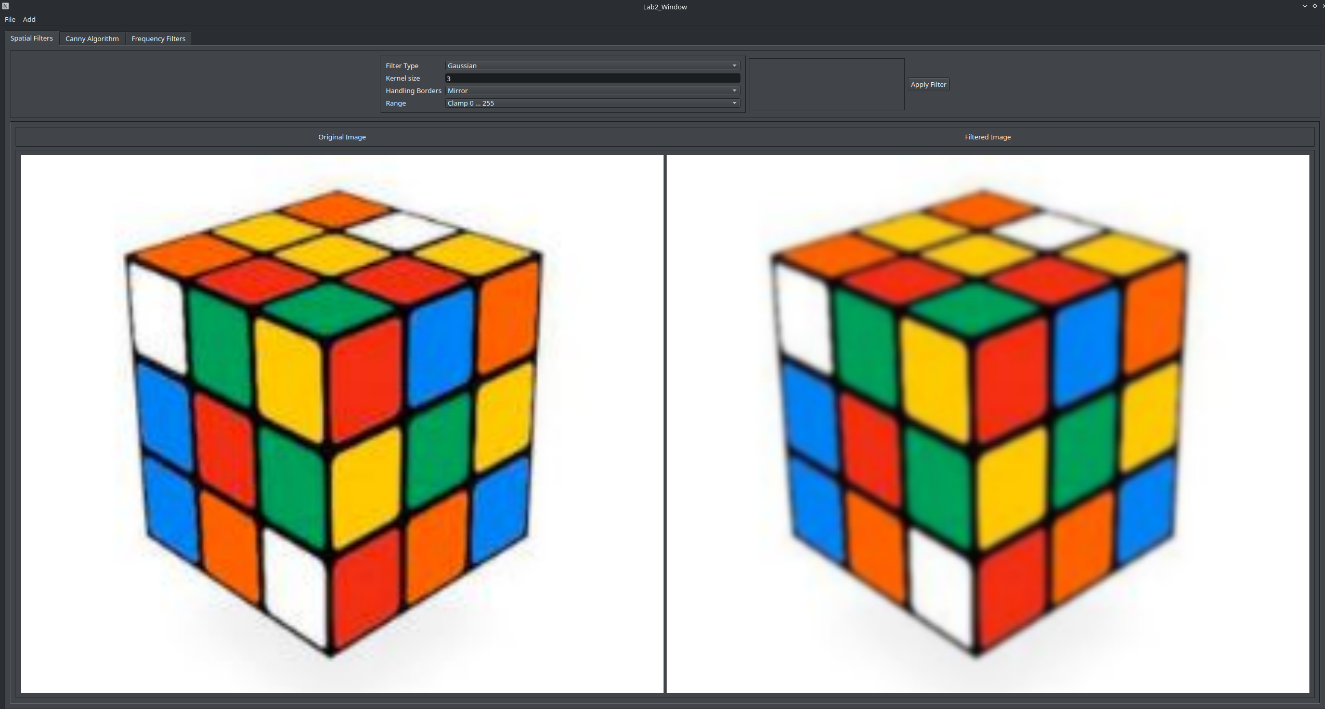
## Résultats

### Spatial Filters

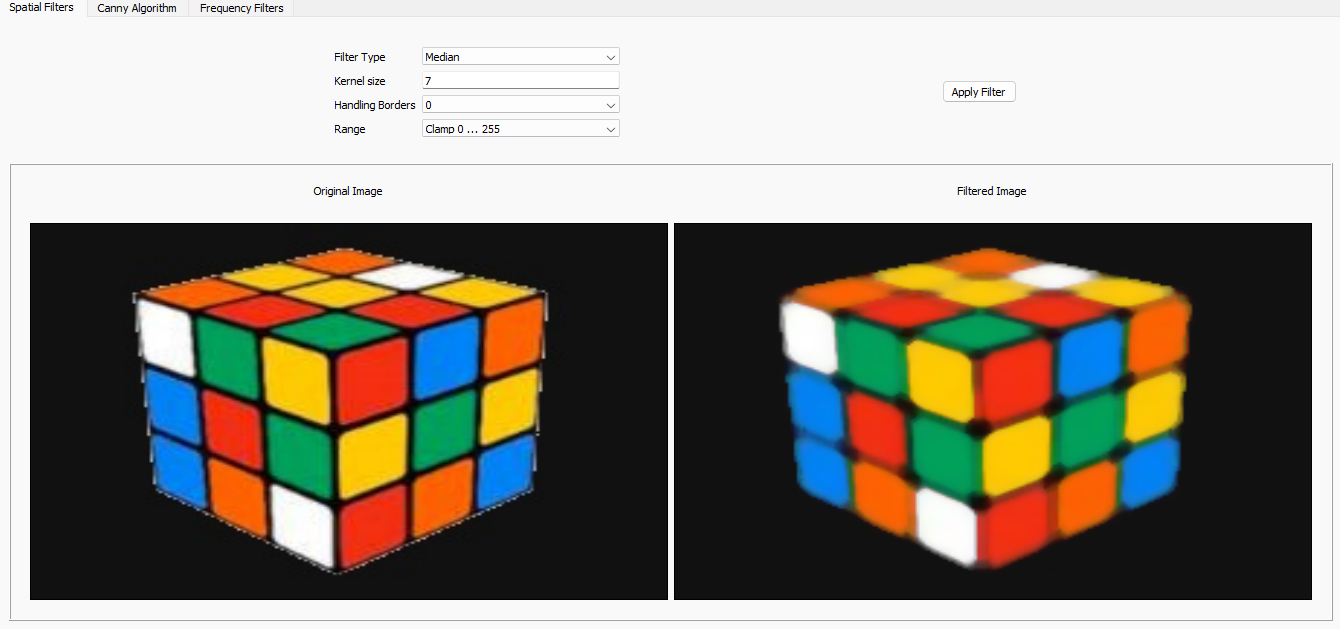
#### Mean

*Figure 1 – Filtre spatial: mean*

#### Gaussian

*Figure 2 – Filtre spatial: Gaussian*

#### Median

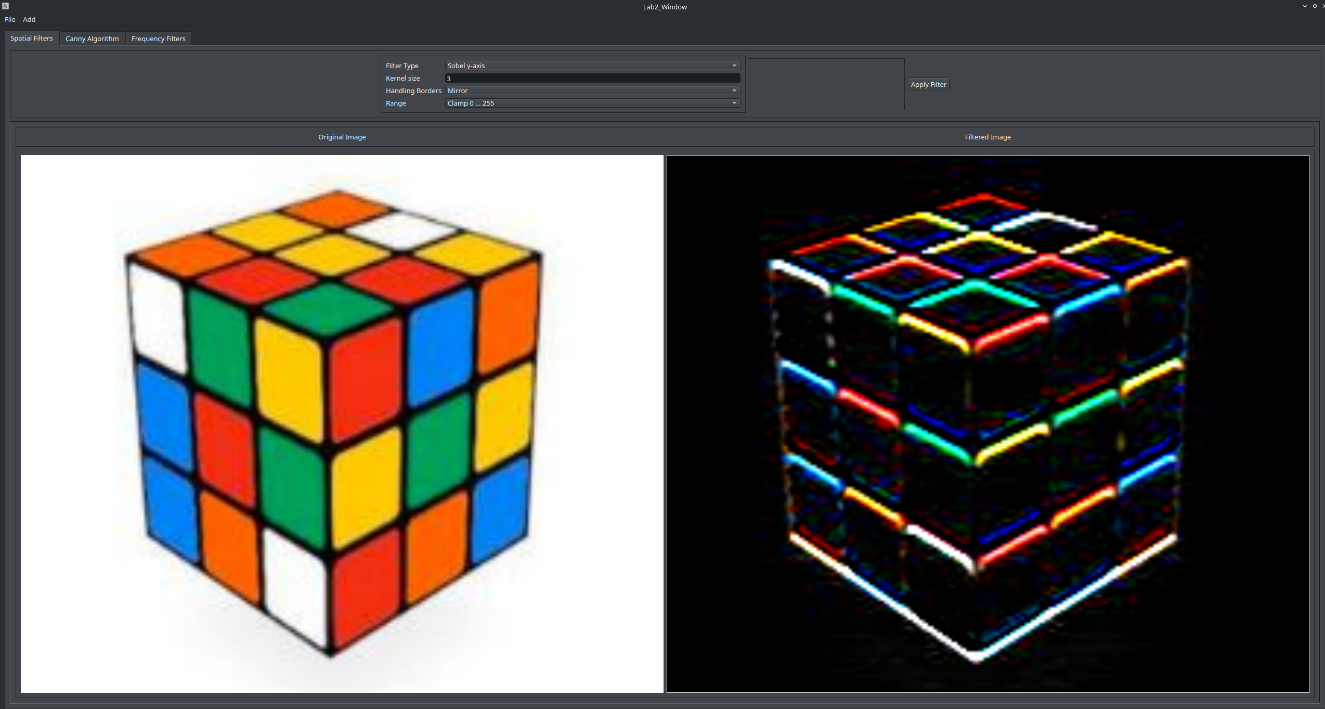
*Figure 3 – Filtre spatial: median*

#### Sobel-x

*Une image contenant Casse-tête mécanique, Rubik's Cube, puzzle, cube

Description générée automatiquementFigure 4 – Filtre spatial: sobel-x*

#### Sobel-y

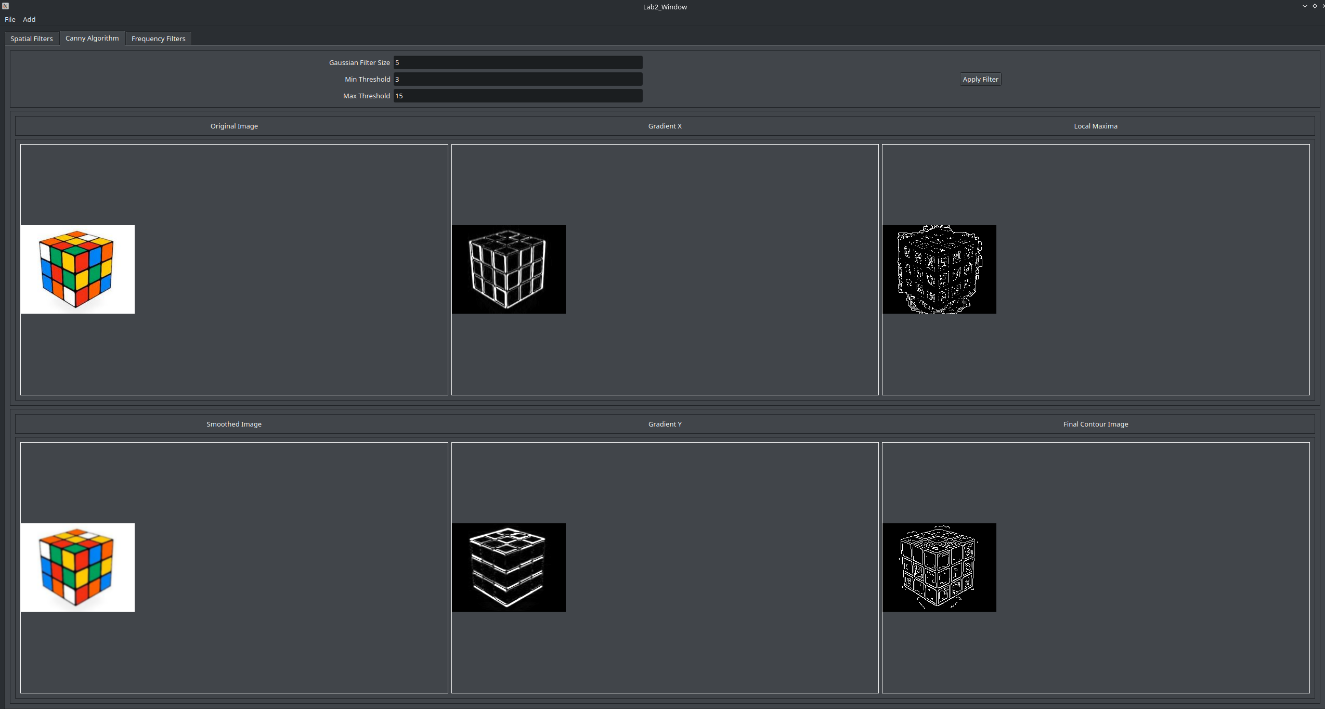
*Figure 5 – Filtre spatial: sobel-y*

#### Sobel

*Une image contenant cube, Casse-tête mécanique, Rubik's Cube, puzzle

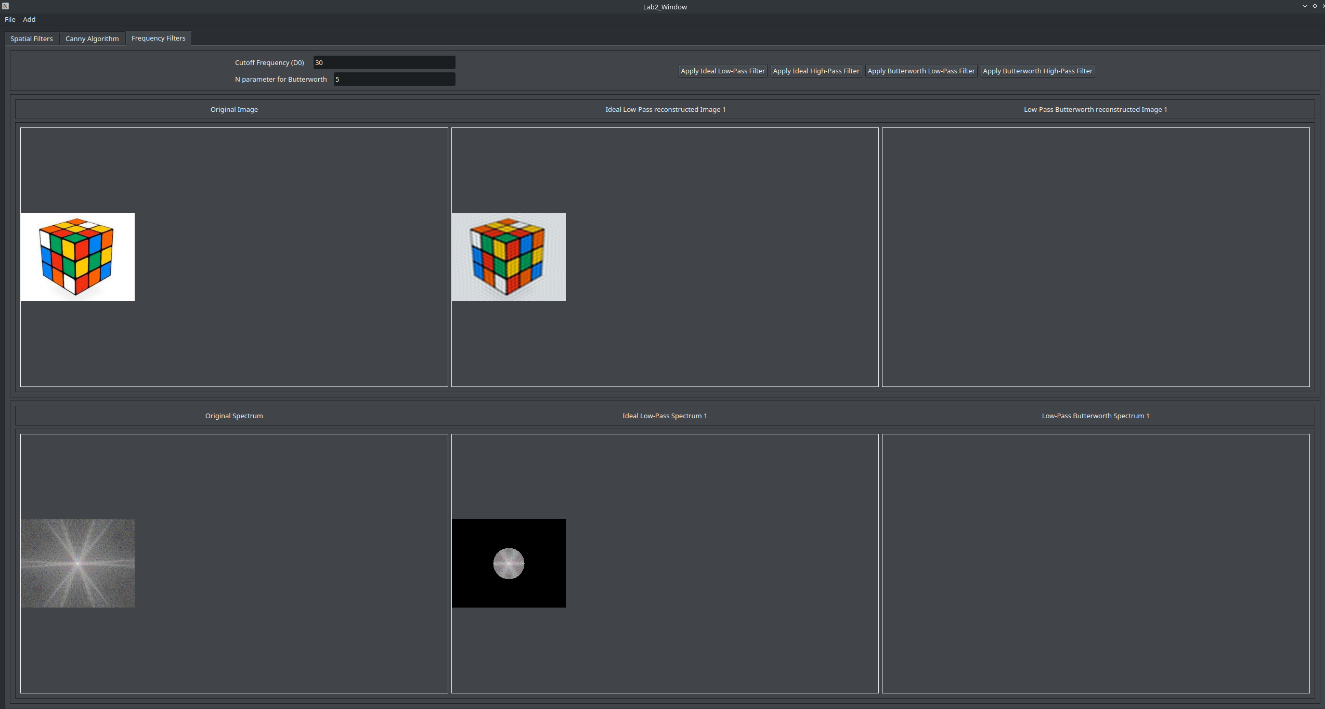
Description générée automatiquementFigure 6 – Filtre spatial: sobel complet*

### Canny algorithm

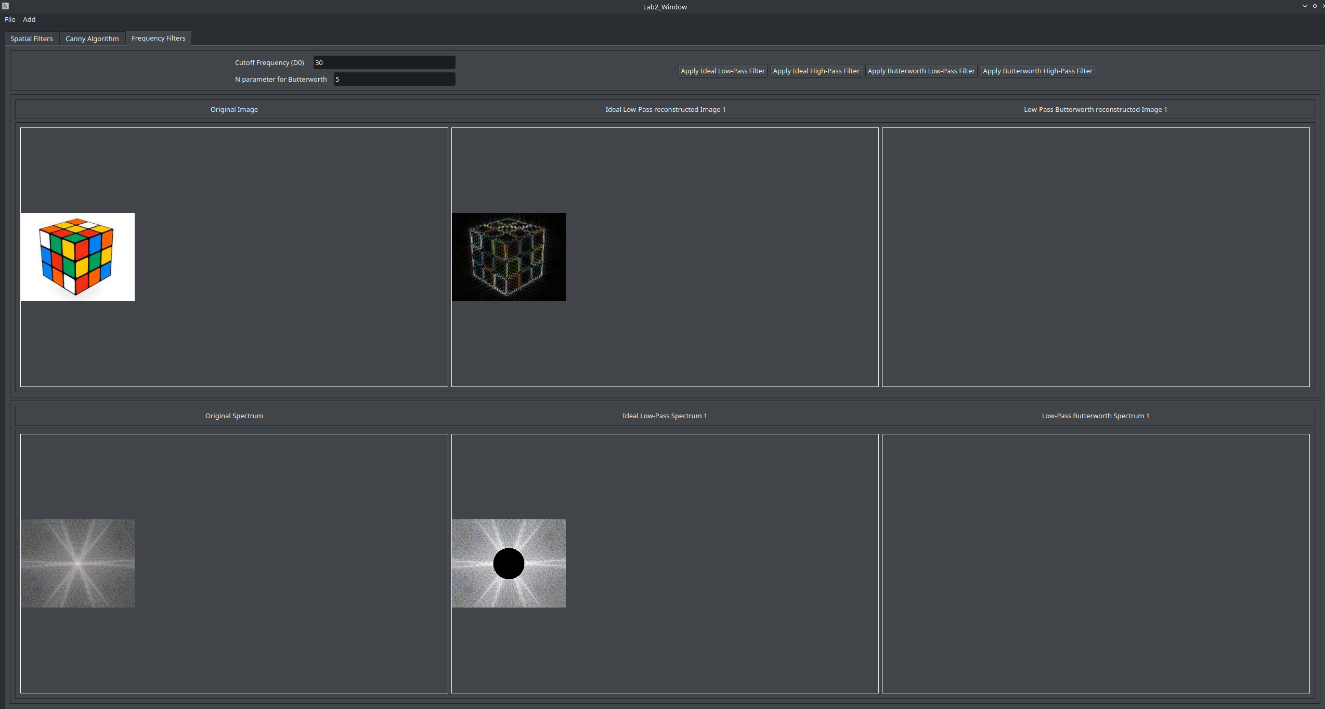
*Figure 7 – Canny algorithm: décomposition de l’algorithme de Canny*

### Frequency Filters

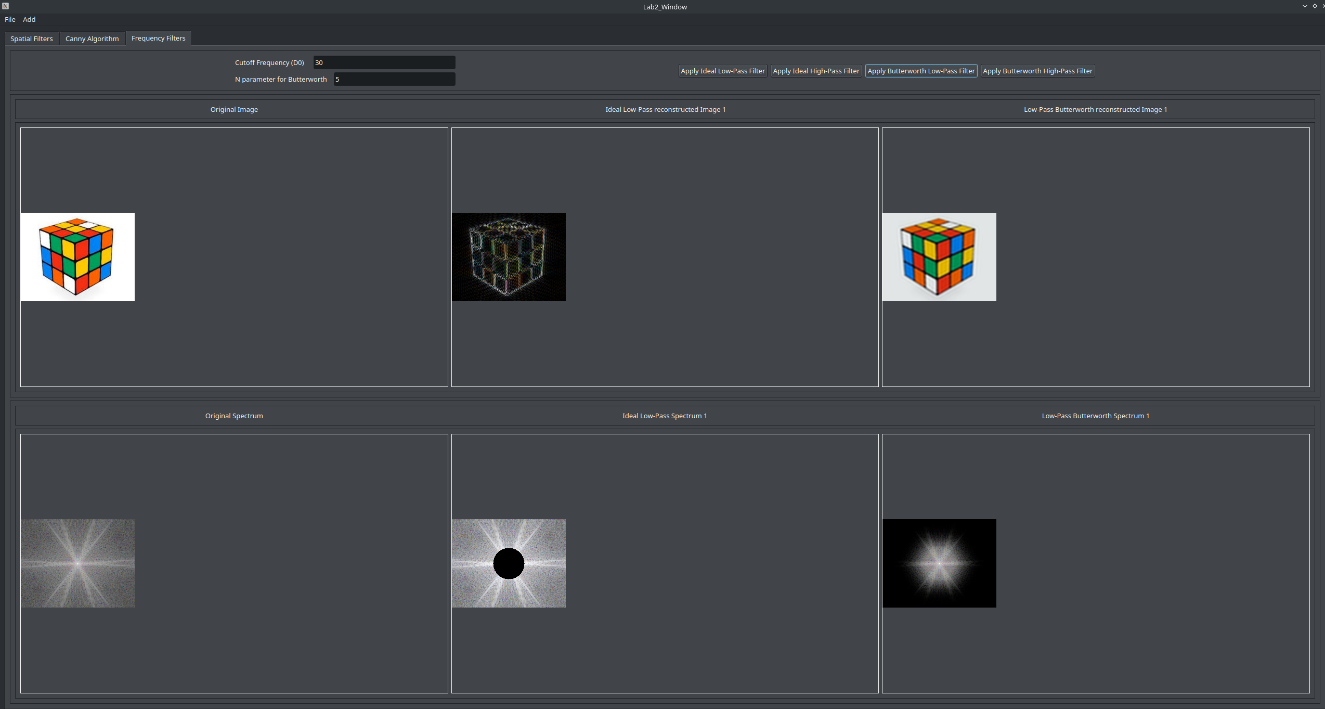
#### Passe-bas idéal

*Figure 8 – Filtre de fréquence: passe-bas idéal*

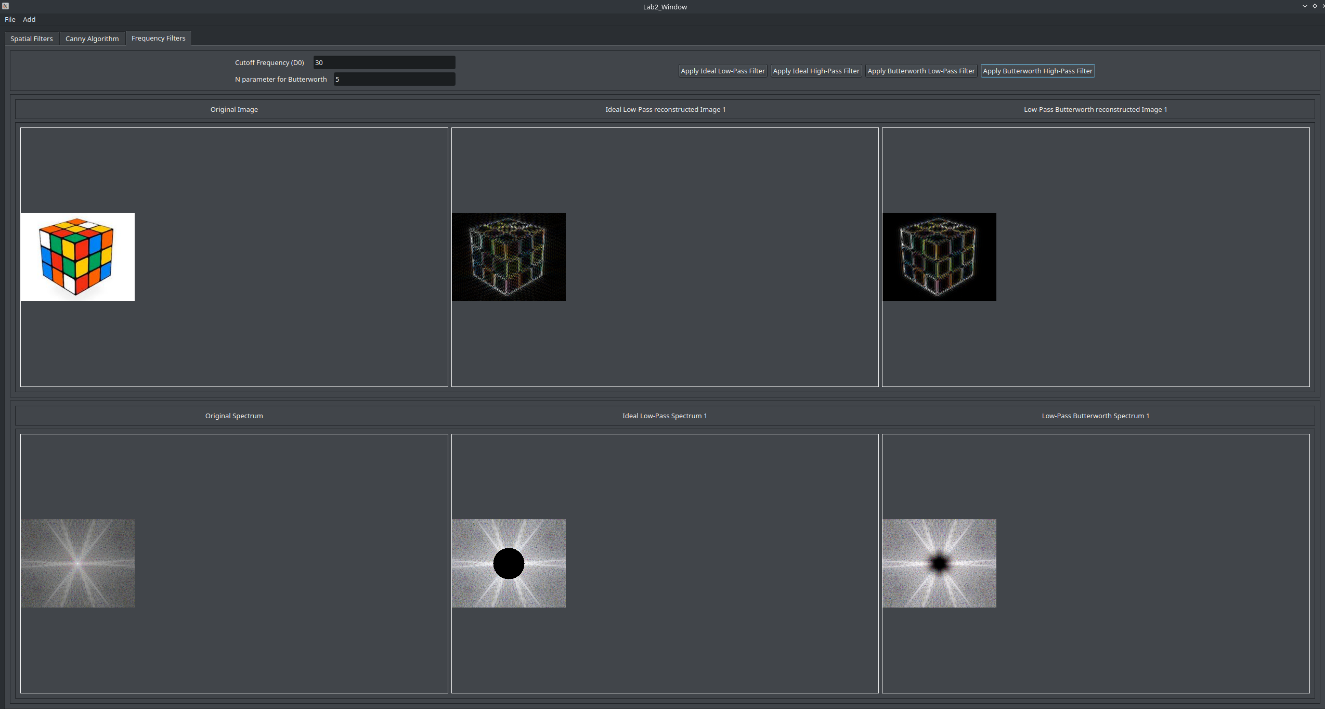
#### Passe-haut idéal

*Figure 9 – Filtre de fréquence: passe-haut idéal*

#### Butterworth passe-bas

*Figure 10 – Filtre de fréquence : passe-bas Butterworth*

#### Butterworth passe-haut

*Figure 11 – Filtre de fréquence : passe-haut Butterworth*

## Difficultés rencontrées

Nous avons rencontré certaines difficultés durant ce laboratoire comme notamment dans la partie 3 sur comment procéder afin de faire les transformée de Fourier (et les transformés inverses de Fourier). C’était une partie complexe comme les commandes Python peuvent être mélangeant et son complexe à appliquer. Ce sont des concepts qui restent très théoriques et les mettre en pratique est souvent plus compliqué. Nous avons tout de même réussi cette partie en divisant en petites fonctions concises chaque étape des filtres qui étaient à faire.

Nous avons identifié un problème dans la partie des filtres spatiaux. Lorsque nous sélectionnons un mode de normalisation autre que **Clamp 0...255**, le filtre appliqué semble perdre son effet, et l’image prend directement la valeur du mode de normalisation choisi. De plus, la variable associée aux **étendues** semble revenir à sa valeur initiale, ce qui entraîne l’affichage de l’image originale sans aucun filtre appliqué. Ce problème est particulièrement gênant, car il empêche de visualiser correctement les différentes méthodes de normalisation. Cette anomalie est probablement due à une mauvaise interprétation du code ou à la manière dont nous avons implémenté l’application des filtres. Bien que le code du modèle soit présent et fonctionnel, il ne semble pas correctement affiché à l’utilisateur, ce qui suggère que l’anomalie provient plutôt de la vue que du modèle. Ce point reste à corriger pour garantir une meilleure visibilité des transformations appliquées.

## Améliorations potentielles

Une amélioration que nous pourrions apporter est la gestion des couleurs dans la partie 3. Les différents filtres demandés ne prennent pas nécessairement en compte les 3 canaux de couleurs, de la manière que les différents modules fonctionnent et comment le code est monté cela fait en sorte que lors de la reconstruction de l’image lors de la transformé inverse de Fourier est fait, les couleurs peuvent être légèrement différente comparativement à l'image originale.

Concernant le problème des filtres spatiaux, une piste d’amélioration serait de s’assurer que l’image filtrée est correctement mise à jour après l’application du mode de normalisation. Une solution possible serait d’appliquer la normalisation **après** le filtrage et non simultanément. Il serait aussi pertinent de vérifier si la variable **range\_method et filter\_method** sont bien mises à jour avant l’affichage de l’image filtrée, pour éviter qu’elle ne reprenne sa valeur initiale.

# Conclusion

Ce laboratoire nous a permis de comprendre les principes fondamentaux du filtrage en traitement d'images, en explorant les différentes techniques dans les domaines spatial et fréquentiel. Nous avons appris à implémenter des filtres classiques tels que gaussien, Sobel et médian dans l’espace spatial, ainsi qu’à ajuster les paramètres du filtre de Canny pour une détection de contours précise. La manipulation d’images dans le domaine fréquentiel, en appliquant des filtres passe-bas et passe-haut, a également été un aspect clé de ce laboratoire, offrant une approche alternative pour améliorer la qualité des images. En somme, cette expérience pratique a enrichi notre compréhension des outils et des algorithmes utilisés dans le traitement d’images, tout en démontrant leur importance dans des applications réelles comme la vision par ordinateur et l’imagerie médicale.

# Annexe : Manuel d’utilisateur

**Installation des dépendances**

Pour installer les différentes librairies, on peut utiliser la commande suivant (avec l’utilitaire *pip*):

pip install numpy opencv-python matplotlib PyQt5

## Exécution du code

On exécute dans le terminal la commande Python suivante:

python Lab\_1\_imagerie/GTI411 - LAB 2/MainWindow.py