Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Automatique
Automatique et informatique industrielle

Réf. : Entrez la référence du document

Présenté et soutenu par :
NEDJAR Hatem

Le : dimanche 26 juin 2022

Traitement d'images en utilisant la plateforme 'GOOGLE COLAB'

Jury :

Dr.	MEGHERBI Hassina	Pr	Université de Biskra	Président
Dr.	ZITOUNI Othmane	MCA	Université de Biskra	Examinateur
Dr.	MESSAOUDI Abdel Hamid	MCA	Université de Biskra	Rapporteur

Année universitaire : 2021-2022

Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Automatique
Automatique et informatique industrielle

Réf. : Entrez la référence du document

Traitement d'images en utilisant la plateforme 'GOOGLE COLAB'

Le : dimanche 26 juin 2022

Présenté par :

NEDJAR Hatem

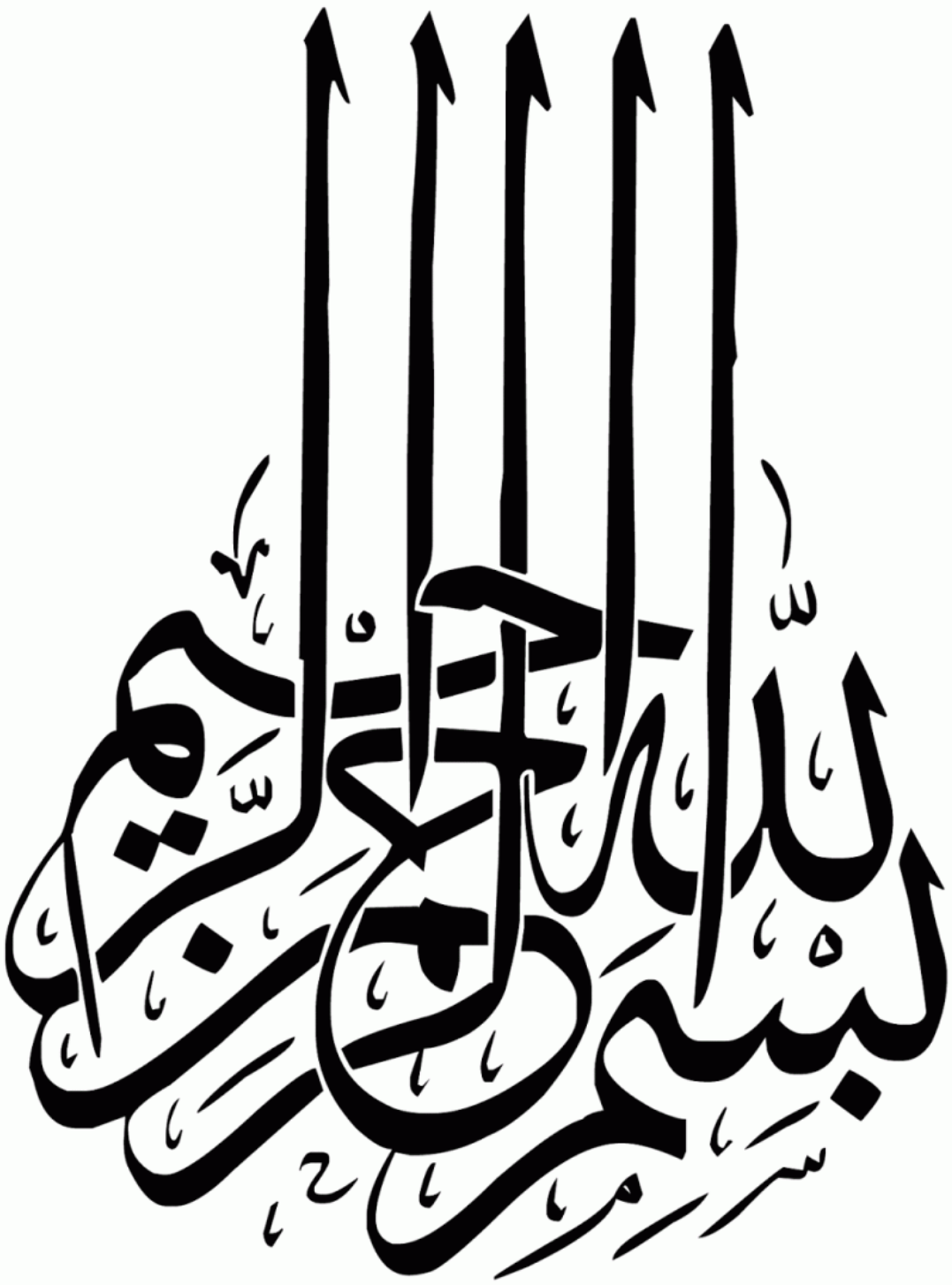
Avis favorable de l'encadreur :

MESSAOUDI Abdel Hamid

Signature Avis favorable du Président du Jury

MEGHERBI Hassina

Cachet et signature


بسم الله الرحمن الرحيم

Remerciements

Je remercie dieu de m'avoir donné la volonté et le courage afin d'arriver à achever ce modeste travail.

Je tiens à remercier vivement mon encadreur Monsieur : ABDELHAMID MESSAOUDI pour sa confiance, ses encouragements, ses précieuses corrections et pour les conseils qu'il a apporté pour l'achèvement de ce projet.

Je tiens également à remercier l'ensemble de membres de jury qui nous ont fait l'honneur de juger notre travail.

Je présente aussi mes remerciements à tous mes professeurs de la spécialité Génie Electrique.

Je tiens aussi à exprimer mes remerciements à tous ceux qui m'a aidé de près ou de loin durant l'élaboration de ce mémoire de fin d'étude.

Dédicaces

Je dédie ce modeste travail aux propriétaires de crédités après "Dieu":

A mon cher père et ma chère mère qui m'ont encouragé et qui ont toujours cru en moi.

A mes frères et leurs enfants.

A toute la famille NEDJAR.

A tous mes amis.

A tous mes formateurs pour leurs efforts et leur amabilité et tous qui m'ont aidé de près ou de loin pour la réalisation de ce travail.

A celle qui partage mes peines et mes joies.

RESUME

Le filtrage et le rehaussement d'images est un champ important qui est employé infiniment dans le traitement d'images, l'image constitue l'un des moyens les plus importants utilisé dans le domaine de communication. L'image peut être exposée à du bruit qui ralentit le processus de communication en raison de son manque de clarté, il faut donc utiliser des méthodes de filtrage pour récupérer les informations et traiter l'image. Dans notre étude, nous utilisons plusieurs filtres (filtres des lissages et filtres des contours) sur les photos bruitées pour obtenir les meilleurs résultats.

Mots clés : traitement d'image, bruit, filtrage, image numérique.

ملخص

تصفية وتحسين الصور مجال مهم يستخدم بكثرة في معالجة الصور، فالصورة من أهم الوسائل المستخدمة في مجال الاتصال. قد تتعرض الصورة للضوضاء مما يؤدي إلى إبطاء عملية الاتصال بسبب افتقارها إلى الوضوح ، لذلك يجب استخدام طرق التصفية لاسترداد المعلومات ومعالجة الصورة. في دراستنا، نستخدم العديد من المرشحات (مرشحات التنعيم ومرشحات توضيح الخطوط) على الصور الصاخبة لنحصل على أفضل النتائج.

الكلمات المفتاحية : معالجة الصور، ضوضاء، ترشيح ، صورة رقمية.

ABSTRACT

Filtering and enhancement of images is an important field which is used infinitely in image processing, the image is one of the most important means used in the field of communication. The image can be exposed to noise that slows down the communication process because of its lack of clarity, so it is necessary to use filtering methods to recover the information and process the image. In our study, we use several filters (smooth filters and outline filters) on the noises photos to obtain the best results.

Keywords: image processing, noise, filtering, digital image.

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Liste des tableaux

Table des matières

Introduction générale

Le traitement d'images commence à être étudié dans les années 1920 pour la transmission d'images, Son développement a commencé au début des années 70, et il a connu récemment un fort développement dû à l'émergence de la haute technologie dans le domaine de l'image et de l'imagerie. [1]

L'image est une représentation visuelle de quelque chose à travers laquelle on peut communiquer avec les autres et se comprendre à travers elle, c'est aussi le moyen de communication le plus efficace, et chacun peut à sa manière, obtenir des impressions et extraire des informations précises. Par conséquent, le traitement d'image est un ensemble de méthodes et de techniques qui opèrent dessus. Ceux-ci peuvent améliorer l'aspect visuel de l'image et extraire informations jugées pertinentes.

Le domaine de traitement d'image comprend également un nombre important de filtres qui peuvent être classés en fonction de leur utilisation : les filtres correcteurs (accentuation, lissage, réduction ou augmentation de bruit, etc.), les filtres de retouche et les filtres d'effets spéciaux.

Dans ce mémoire, il s'agit d'un problème qui se concentre sur les étapes de traitement d'image plus précisément, le filtrage d'images. Le filtrage d'images est une règle de l'informatique et des mathématiques appliquées qui permet d'interpréter image traitée. Le but du filtrage est d'améliorer la qualité de l'image ou d'extraire informations. Le travail de ce mémoire consiste à introduire les formulations principales mathématiques et informatiques pour éliminer le bruit et la détection des contours pour segmenter les objets présents image.

Le présent mémoire se décompose en trois chapitres :

Dans **le premier chapitre,** on introduit les concepts généraux de l'image et du traitement d'image.

On fournit dans **le deuxième chapitre** les notions générales sur le bruit et les filtres utilisés en filtrage.

Dans **le troisième chapitre**, on présente le travail réalisé et les résultats obtenus.

Nous terminons ce mémoire par une conclusion générale et des perspectives.

Chapitre I Généralités sur le traitement d'images

I.1. Introduction

Le traitement d'images est un domaine très vaste qui a connu et connaît encore un développement important au cours des dernières décennies. Le traitement numérique des images désigne l'ensemble des techniques permettant de modifier les images numériques pour les améliorer ou en extraire des informations.

Ainsi, le traitement d'images est un ensemble de méthodes et de techniques permettant de les manipuler afin de rendre de telles opérations possibles, plus simples, plus efficaces et plus agréables pour améliorer l'effet visuel des images et en extraire des informations pertinentes.

Dans ce chapitre, nous présenterons les concepts de base nécessaires à la compréhension des images numériques et des systèmes de traitement.

I.2. Généralités sur le traitement d'images

I.2.1. Définition de l'image

L'image est une représentation d'une personne ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film, etc. C'est aussi un ensemble structuré d'informations qui, après affichage sur l'écran, ont une signification pour l'œil humain.

L'image peut être décrite sous la forme d'une fonction I(x,y) de brillance analogique continue, définie dans un domaine borné, Les variables x et y sont les coordonnées spatiales d'un point de l'image et I est une fonction d'intensité lumineuse et de couleurs. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation [2].

I.2.2. Image numérique

Contrairement aux images obtenues avec un appareil photo ou dessinées sur papier, les images traitées par ordinateur sont numériques (représentées par une suite de bits).

Une image numérique est une image dont la surface est divisée en éléments de taille fixe appelés cellules ou pixels, chaque élément étant caractérisé par une échelle de gris ou un niveau de couleur pris à l'endroit correspondant dans l'image réelle [2].

Une image numérisée est convertie d'un état analogique en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x,y), comme illustré à la figure I.1.

Pour des commodités de présentation et d'adressage, les données d'image sont généralement agencées sous la forme d'un tableau I à n lignes et p colonnes.

Chaque élément I(x, y) représente un pixel de l'image, et sa valeur est associée au niveau de gris codé sur m bits

(2^m niveaux de gris ; 0 = noir ; 2^m -1 = blanc)

La valeur de chaque point représente la mesure d'intensité lumineuse perçue par le capteur.

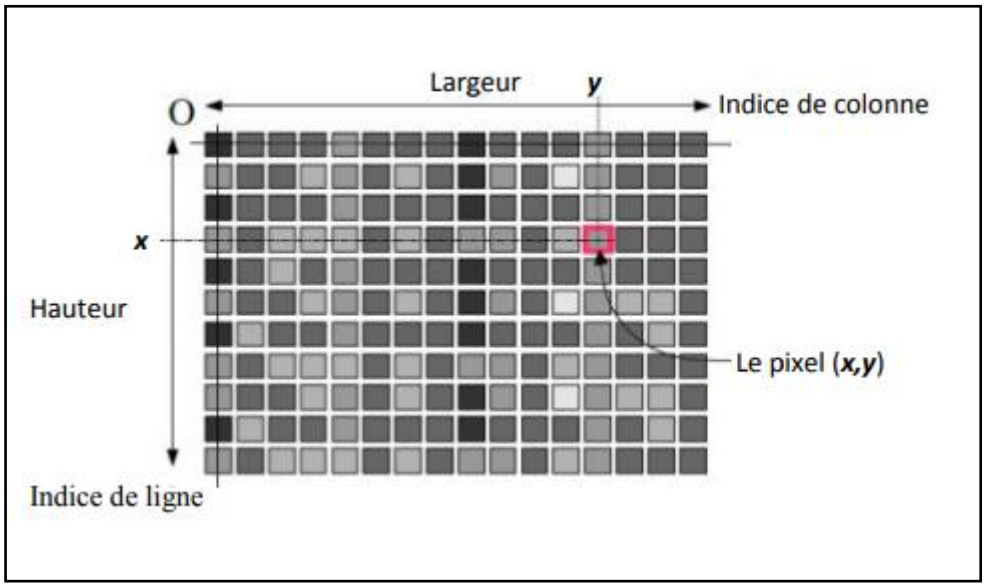


Figure I.1 : Représentation d'image numérique.

I.2.3. Caractéristiques de l'image numérique

I.2.3.1. Pixel

Le pixel, abréviation du terme "élément d'image", est une unité de surface qui définit la base d'une image numérique. Il matérialise un point donné (x, y) du plan image. L'information représentée par un pixel est le niveau de gris (ou de couleur) obtenu à partir de l'emplacement correspondant dans l'image réelle. La différence entre une image monochrome et une image couleur est la quantité d'information contenue dans chaque pixel, par exemple dans une image couleur (RVB : Rouge, Vert, Bleu) la valeur du pixel de chaque couleur est représentée par trois octets. [3]

I.2.3.2. Dimension

La dimension de l'image correspond àsa taille. Une image se présente sous la forme d'une matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes dans cette matrice multipliée par le nombre de colonnes donne le nombre total de pixels dans l'image.

I.2.3.3. Résolution

Il s'agit de la netteté ou des détails fins qu'un moniteur ou une imprimante peut atteindre lors de la production de l'image. Sur les écrans d'ordinateur, la résolution est exprimée en pixels par unité de mesure (pouces ou centimètres). Nous utilisons également le mot résolution pour désigner le nombre total de pixels sur un écran pouvant être affiché horizontalement ou verticalement. Plus le nombre est élevé, meilleure est la résolution. La résolution d'une image correspond au niveau de détails qui sera représenté sur cette image.

Pour la numérisation, les deux équations suivantes doivent être considérées :

$$(X * résolution) = x\ pixels \tag{I.1}$$

$$(Y * résolution) = y\ pixels \tag{I.2}$$

Telle que :

- X et Y représentent la taille (pouce ou cm, 1 pouce=2,54 centimètres) de la structure à numériser.
- Résolution représente la résolution de numérisation.
- x et y représentent la taille (en pixels) de l'image.

I.2.3.4. Contours et textures

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contours consiste à identifier dans l'image les points qui séparent deux textures différentes. [4]

Figure I.2 : Contours d'une image.

I.2.3.5. Bruit

Le bruit (parasites) dans une image est considéré comme un phénomène dans lequel l'intensité d'un pixel change soudainement par rapport à ses voisins, et il provient de l'éclairage de l'optique et de l'électronique du capteur. [5]

(a) (b)

Figure I.3 : Image Makam echahid, a) Image sans bruit et b) image avec bruit

I.2.3.6. Luminance

C'est le degré de la luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface. Généralement, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.

Une bonne luminance se caractérise par [6]:

- des images lumineuses (brillantes)
- un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

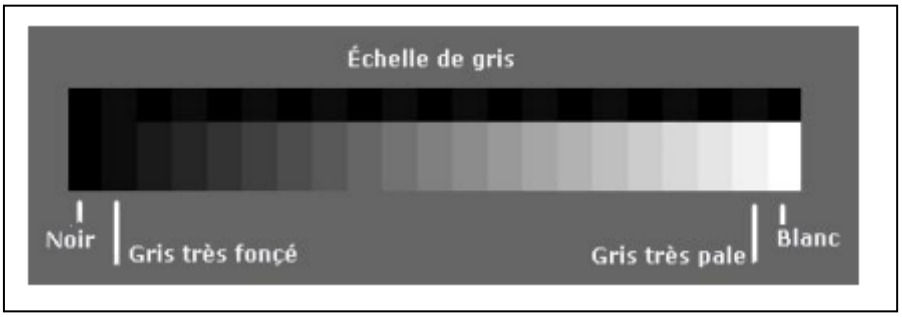


Figure I.4 : Variations de luminosité

I.2.3.7. Contraste

Le contraste est généralement défini comme l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de l'image. Le contraste est défini en fonction des luminances de deux zones d'images.

Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport : [7]

$$C = \frac{L1-L2}{L1+L2} \qquad \text{(I.3)}$$

I.2.3.8. Histogramme

L'histogramme de niveaux de gris ou de couleurs d'une image est une fonction qui indique la fréquence à laquelle chaque niveau de niveaux de gris (couleur) apparaît dans l'image. L'histogramme est généralement modifié pour réduire les erreurs de quantification, pour comparer deux images obtenues sous un éclairage différent ou pour mesurer une propriété d'une image.

Il fournit de nombreuses informations sur la distribution des niveaux de gris (couleurs) et voir entre quelles limites la distribution des niveaux de gris (couleurs) se situent dans le cas d'une image trop claire ou d'une image trop sombre. Il permet d'améliorer la qualité d'une image en introduisant certaines modifications afin d'en extraire des informations utiles. [4]

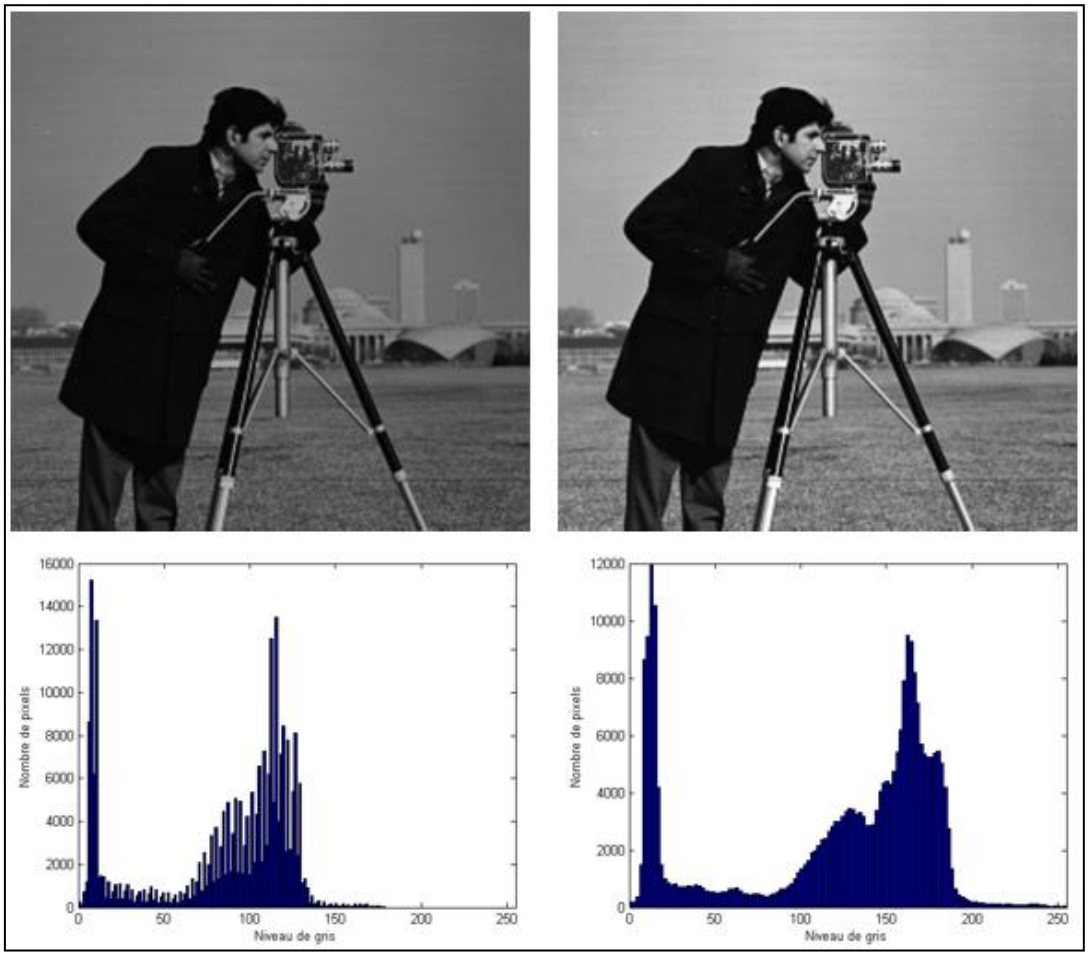


Figure I.5 : Image cameraman avec son histogramme.

I.2.3.9. La taille d'une image

La taille de l'image est le nombre des pixels que multiplie sa taille en octet. [2]

Exemple : pour une image de 240×420 en TrueColor :

- Nombre de pixels : $240 \times 420 = 100800$ pixels.
- Taille de chaque pixel : $\frac{24 \text{ bits}}{3} = 3$ octets
- Le poids de l'image est ainsi égal à : $100800 \times 3 = 302400 \text{ bits} = \frac{302400}{1024} = 295\text{Ko}$

I.2.4. Types d'images numériques

Généralement, il existe deux types d'images numériques qui utilisent des technologies complètement différentes et sont adaptées à des usages tout aussi différents. Il s'agit des images matricielles et des images vectorielles. [8]

La figure I.6 vous montre une des différences entre ces deux types d'images avec un zoom $\times 3$ sur un détail. Le contour du fruit est parfaitement lisse dans le cas de l'image vectorielle et apparaît légèrement « cranté » dans le cas de l'image matricielle.

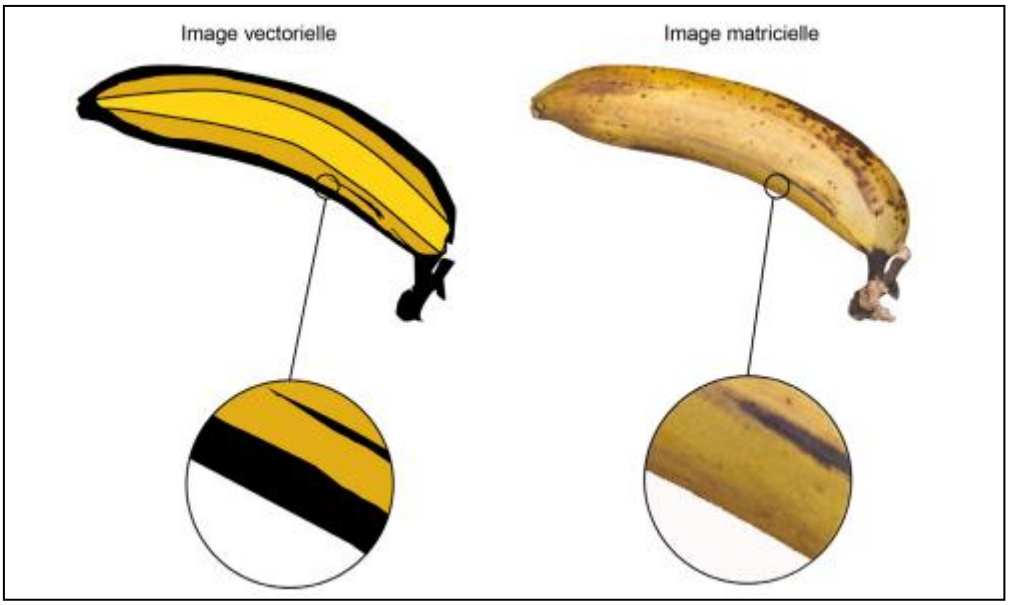


Figure I.6 : Exemple d'une image vectorielle et matricielle

I.2.4.1. L'image vectorielle

Une image vectorielle est constituée d'objets "simples" décrits par une fonction, telle que f(x)=ax+b, qui correspond à une droite faisant un angle avec l'horizontale. Chaque objet est décrit individuellement par une fonction.

Ce type d'image est idéal pour créer des dessins, des illustrations, des logos et des pictogrammes. L'un des avantages des images vectorielles est qu'elles peuvent être agrandies ou réduites à volonté sans perte de qualité ni augmentation de la taille du fichier. [8]

I.2.4.2. L'image matricielle

L'image matricielle ou Bitmap est utilisée pour les captations de photo, de vidéo, de numérisation de documents. Une image matricielle est composée de petits carrés d'une seule couleur chacun, appelés pixels, rangés en ligne et en colonne, sous forme de matrice. Le pixel est l'élément de base d'une image matricielle. [8]

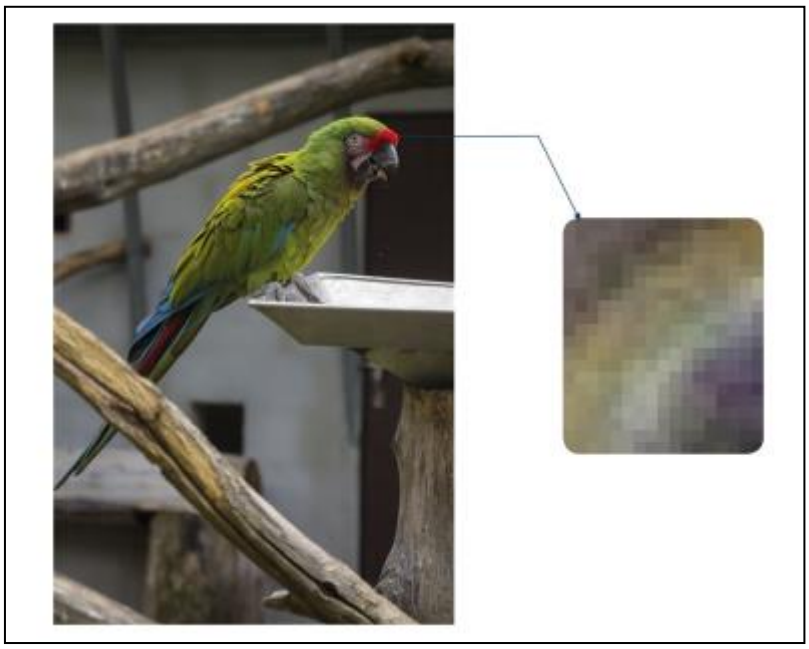


Figure I.7 : Exemple d'image matricielle avec un zoom.

I.2.5. Codages des couleurs

Généralement, on distingue trois grands types d'images numérique :

- Image noir et blanc,
- Image niveaux de gris,
- Image couleur.

Ces types sont généralement à choisir lors d'une numérisation par scanner ou lors de la configuration d'un appareil photographique. [9]

I.2.5.1. Image noir et blanc

Le contenu de chaque case de la matrice de l'image est soit un 0 (noir) soit 1 (blanc). Le nombre de couleurs n'est que de 2 mais parfois suffisant dans le cadre par exemple de documents scripturaux. [9]

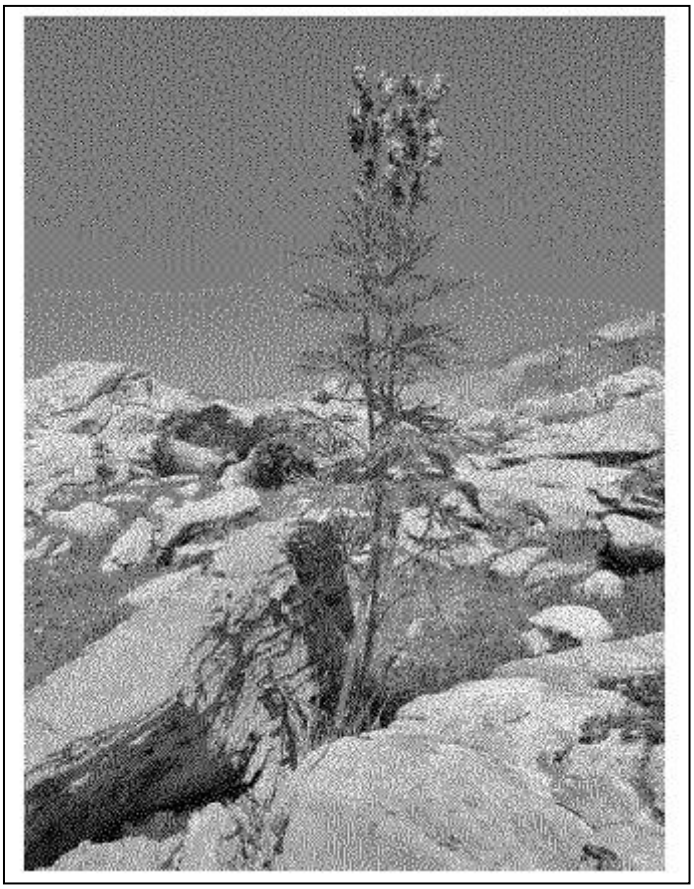


Figure I.8 : Image noir et blanc

I.2.5.2. Image niveaux de gris

Le codage dit en niveaux de gris permet d'obtenir plus de nuances que dans l'image noir et blanc. Il offre des possibilités supplémentaires pour coder le niveau de l'intensité lumineuse. La couleur du pixel est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d'obtenir 256 niveaux de gris (0 pour le noir et 255 pour le blanc). Il existe aussi le codage à 16 niveaux de gris (4 bits). [9]

L'utilisation de ce type de codage (niveaux de gris) est souvent utilisée pour l'impression papier ou pour l'envoi par courrier électronique de fichiers image de taille réduite tout en réduisant la perte de lisibilité de l'image.

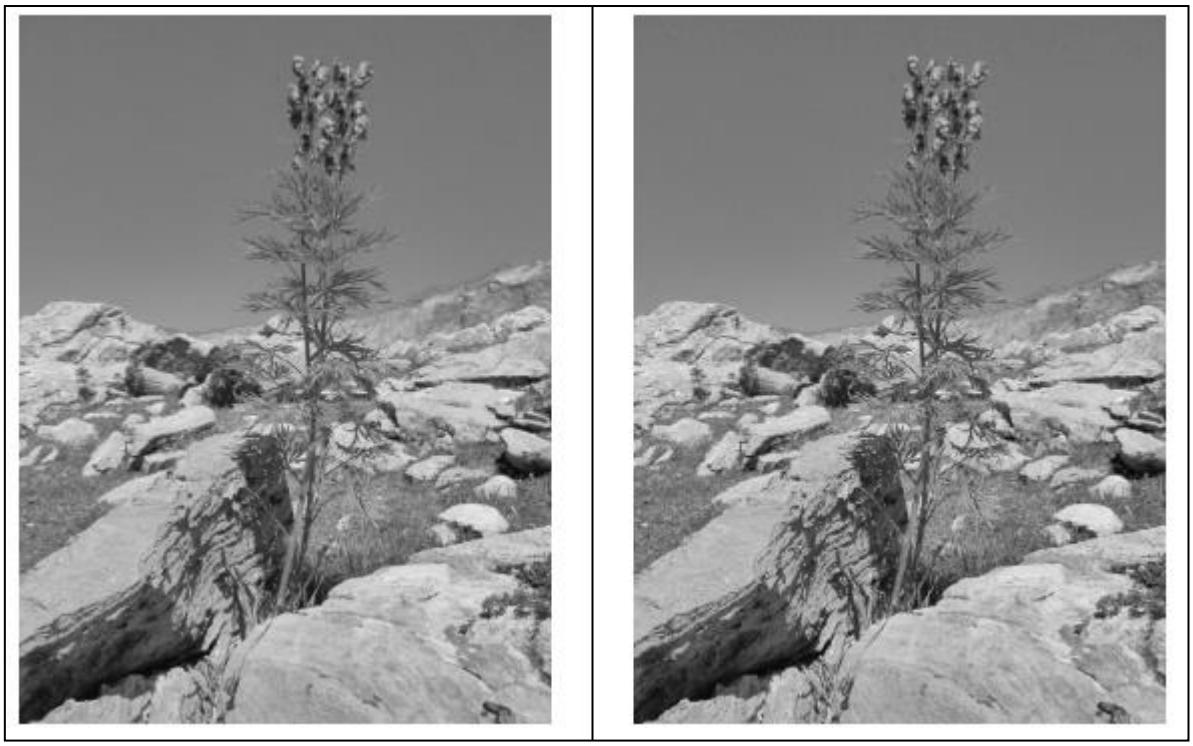


Figure I.9 : Image 'fleur' : avec 256 niveaux de gris (taille : 336 ko) et avec 16 niveaux de gris (taille : 170 ko)

I.2.5.3. Image couleur

a. Codage RVB

Le principe de ce codage consiste à mélanger les trois couleurs : Rouge, Vert et Bleu (noté RVB ou RGB en anglais). A l'aide de ces trois couleurs, on obtient toute une palette de nuances allant du noir au blanc. A chaque couleur est associé un octet (donc 256 niveaux de luminosité) de chacune des couleurs fondamentales. [9]

Un pixel 'couleur' est alors codé avec trois octets et on obtient alors 2^{24} possibilités de couleurs, soit environ 16 millions couleurs différentes. L'image résultante est appelée "vraie couleur" "True Color". La qualité colorimétrique obtenue est celle d'une photographie argentique couleur.

Rouge	Vert	Bleu	Couleur
0	0	0	Noir
0	0	1	Nuance de noir
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
128	128	128	Gris
255	255	255	Blanc

Figure I.10 : Principe codage de la couleur dans le système RVB

Figure I.11 : Image couleur par codage RGB

I.2.6. Système de traitement d'image

Un système de traitement d'image est généralement composé des unités suivantes [3] :

- Un système d'acquisition et de numérisation qui permet d'effectuer l'échantillonnage et la quantification de l'image.
- Une mémoire de masse pour stocker les images numérisées.

- Un système de visualisation.
- Une unité centrale permettant d'effectuer les différentes opérations de traitement d'images.

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation. Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales : les caméras numériques et les scanners. [1]

Tout système de traitement d'images est équipé d'un dispositif d'affichage permettant d'afficher des images. A cet effet, différents types de supports peuvent être utilisés : moniteurs vidéo, plaques photographiques,… [10]

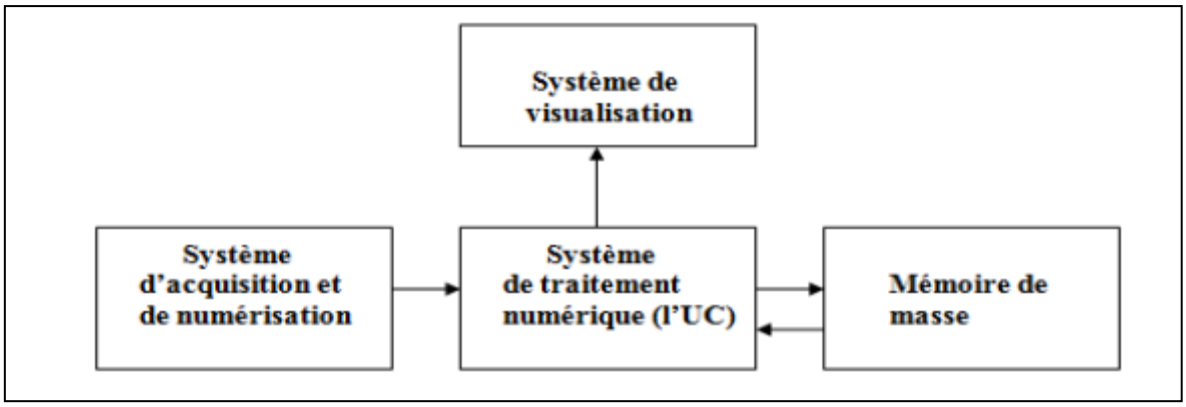


Figure I.12 : Composition d'un système de traitement numérique

I.3. Conclusion

Dans que ce chapitre, on a donné une brève introduction aux concepts liés aux images. L'image est un ensemble structuré de données stocké sous forme de matrice. Chaque élément de cette dernière correspond à un pixel de l'image. Les caractéristiques essentielles de l'image telle que la dimension, la résolution et le type sont liées étroitement à cette matrice. Le chapitre suivant discutera les méthodes de filtrage utilisées en traitement d'images.

Chapitre II Méthodes de filtrages

II.1. Bruit

Le bruit c'est un signal "parasite" dont la distribution dans l'image est aléatoire et la plupart du temps inconnue. La variation soudaine de la couleur d'un pixel par rapport à ses voisins est un phénomène qu'on appelle bruit. Ce dernier peut provenir de l'éclairage ou des dispositifs électroniques du capteur. [5]

II.1.1. Les différentes sources de bruit [11][12]

Les sources de bruit sont vastes et diverse et peuvent provenir si :

- L'image est numérisée à partir d'une photo prise sur film. Le grain du film est la source principale du bruit. Ce dernier peut également être le résultat de l'endommagement du film ou être introduit par le scanner lui-même.
- L'image est acquise directement au format numérique, le mécanisme de collecte des données peut introduire du bruit.
- Des niveaux de lumière et de température du capteur insuffisants peuvent introduire le bruit dans l'image.
- La transmission électronique de données d'image peut introduire bruit.
- Des interférences dans le canal de transmission peuvent également corrompre l'image.
- Des particules de poussière sont présentes sur l'écran du scanner, ils peuvent aussi introduire du bruit dans l'image.

II.1.2. Types de bruit et modélisation

On peut définir le bruit comme étant une dégradation dans l'image, provoquée par une perturbation externe. Si une image est envoyée par voie électronique d'un endroit à un autre, via le satellite ou la transmission sans fil, ou par le câble réseau, on peut s'attendre à des erreurs se produisant dans le signal de l'image. Ces erreurs apparaissent sur l'image de sortie de différentes manières en fonction du type de la perturbation dans le signal.

Généralement, on peut savoir les types d'erreurs à attendre, et donc le type de bruit sur l'image, d'où nous pouvons choisir la méthode la plus adaptée pour réduire les effets. Le filtrage d'une image corrompue par le bruit est donc un domaine important de la restauration d'image.

Les images peuvent être entachées de bruits de nature différente. On s'intéressera ici aux bruits :

- Bruit additif, gaussien,
- Bruit de flou (convolution),
- Bruit poivre et sel.
- Bruit Poisson,
- Bruit Gamma.

Dans ce qui suit, l'image est représentée par**I**, et le bruit (le plus souvent gaussien) par η.

***a.* Bruit additif**

Ces dégradations sont liées au capteur qui provoque par l'erreur de mesure, par fois ces dégradation sont liées à la perturbation de l'environnement comme une tache de café, elles sont modélisées par l'équation :

$$f = KI + \eta \qquad \text{(II.1)}$$

Où :

- **f** image dégradée (observée),
- **K** est un masque.

***b.* Bruit multiplicatif**

Le bruit multiplicatif est un phénomène commun dans tous les systèmes d'imagerie cohérents tels que le laser, l'acoustique, les ultrasons. En général, le bruit multiplicatif est modélisé selon l'expression suivante :

$$f = KI.\eta \qquad \text{(II.2)}$$

***c.* Le flou**

Le type de bruit flou peut provenir d'une erreur de manipulation du matériel d'acquisition telle qu'une mauvaise focalisation ou un bougé. Alors onpeut modeler une image floutée par l'équation :

$$f = KI * \eta \tag{II.3}$$

Où * représente le produit de convolution.

La figure II.1 montre la dégradation d'une image par le flou.

Figure II.1 : Exemples de dégradations d'image : a) image originale, b) image floutée

II.1.3. Exemples de bruits

II.1.3.1. Bruit gaussien (Bruit d'amplificateur)

Le terme modèle de bruit normal est le synonyme de bruit gaussien. Ce modèle de bruit est de nature additive et suit la distribution gaussienne. Cela signifie que chaque pixel de l'image bruyante est la somme de la valeur de pixel vraie et d'une valeur aléatoire de bruit distribué gaussien. Le bruit est indépendant de l'intensité de valeur de pixel à chaque point. [12]

La PDF (Probability Density function) de la variable aléatoire gaussienne est donné par [11]:

$$P(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{II.4}$$

Où P(z) est le bruit de distribution gaussien en image; μ et σ sont respectivement l'écart moyen et l'écart type.

La courbe de bruit gaussien normalisé est en forme de cloche en raison du même caractère aléatoire. La PDF de ce modèle de bruit montre que 70% à 90% des valeurs des pixels brouillés de l'image sont dégradées entre μ-σ et μ+σ. La forme de l'histogramme normalisé est presque la même dans le domaine spectral.

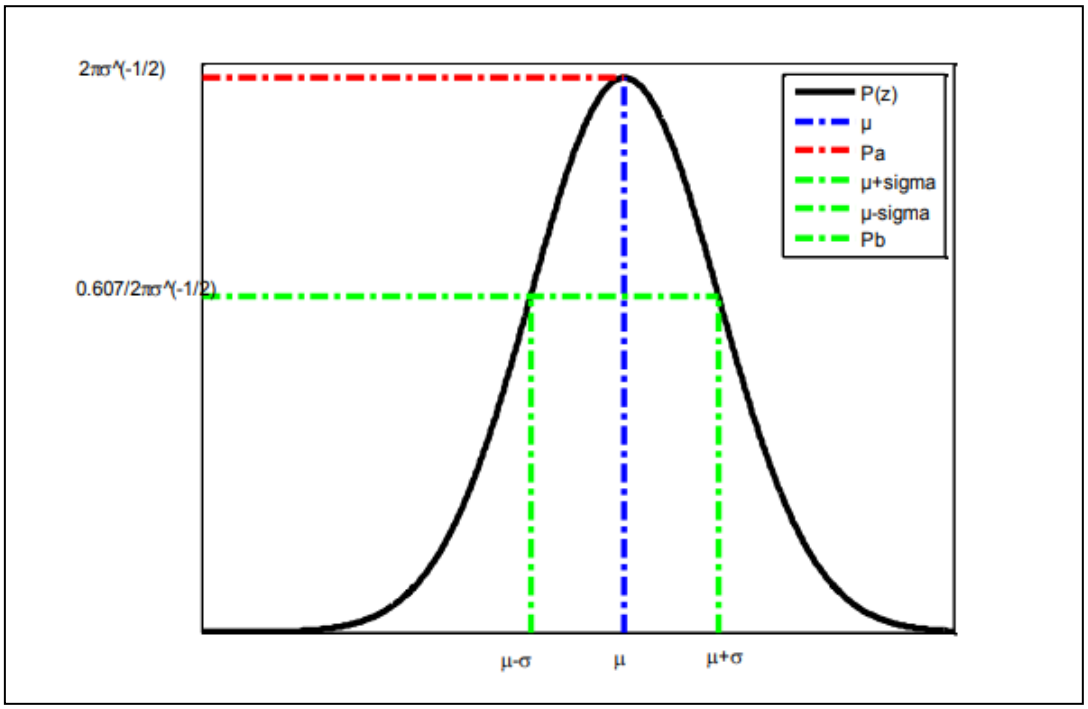


Figure II.2 : Distribution du bruit gaussien (PDF).

II.1.3.2. Bruit sel et poivre

Le bruit de type sel et poivre est un bruit impulsif. Il s'agit en fait des pics d'intensité. Ce type de bruit est dû à des erreurs de transmission de données. Ce bruit se produit dans l'image en raison de changements brusques et soudains du signal d'image. Pour les images corrompues par le bruit sel et poivre, les pixels bruyants ne peuvent prendre que les valeurs maximales et minimales de la plage dynamique. Il s'avère qu'une image 8 bits, la valeur typique pour le bruit poivre est de 0 et pour le bruit sel, elle est de 255. [11][13]

Ce type contient des occurrences aléatoires des valeurs d'intensité de noir et de blanc, et souvent causé par le seuil de l'image bruitée. Le bruit de distribution sel et poivre peut être exprimé par:

$$P(z) = \begin{cases} P_a & \text{pour } z = a \\ P_b & \text{pour } z = b \\ 0 & \text{Ailleurs} \end{cases} \qquad \text{(II.5)}$$

Où Pa, Pb sont les fonctions de densité de probabilités (pdf), P(z) est la distribution du bruit sel et poivre dans l'image.

La figure II.3 présente l'allure de la distribution du bruit poivre et sel.

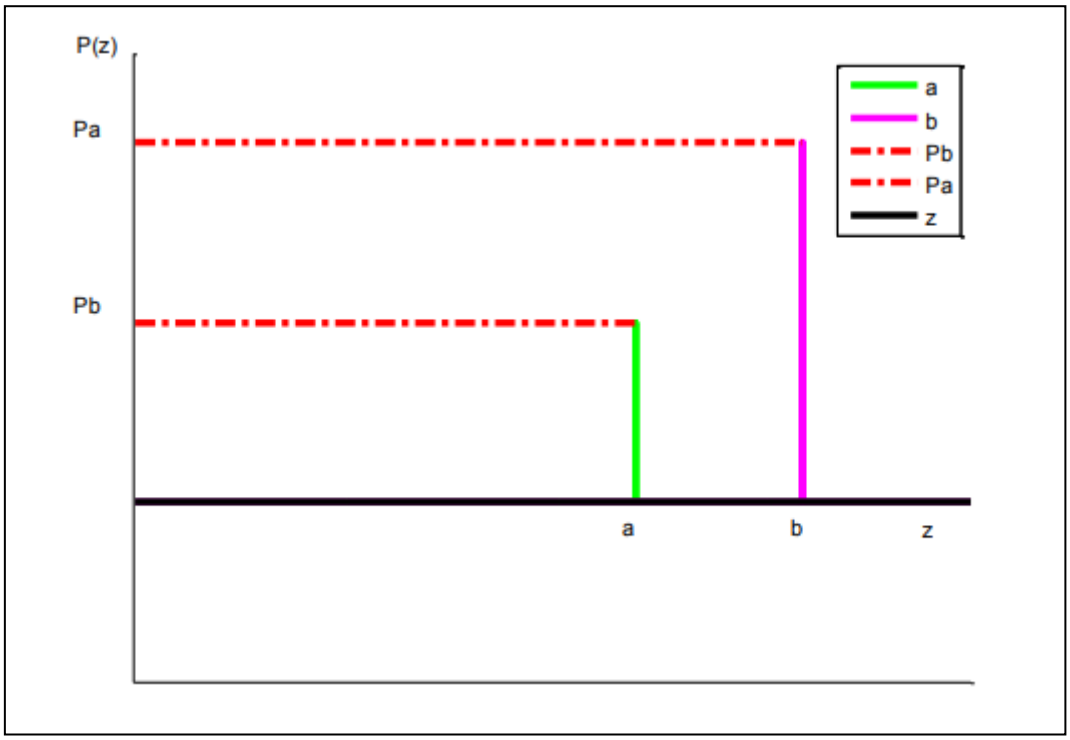


Figure II.3 : Distribution du bruit poivre et sel.

On deux zones, la première est la zone claire (niveau de gris inférieur) appelée "zone a" et l'autre est la zone sombre (niveau de gris supérieur) appelée "zone b".Le Les valeurs du bruit sel et poivre sont les valeurs minimales et maximales dans la "zone a" et la "zone b", respectivement.

Le bruit sel et poivre corrompt souvent les images numériques par des éléments de pixel défectueux dans les capteurs de l'appareil photo, des erreurs d'espace mémoire dans la mémoire, des erreurs de numérisation, etc.

II.1.3.3. Bruit de Poisson

Le bruit de Poisson ou le bruit de projectile est un type de bruit électronique qui se produit quand le nombre fini de particules qui portent l'énergie, telle que des électrons dans un circuit électronique ou des photons dans un circuit optique, est assez petit pour provoquer des fluctuations statistiques discernables dans une mesure.[12]

Ce bruit obéit à la distribution de Poisson et est donné par :

$$P(f_{(pi)}) = k = \frac{\lambda^{k\,i}\, e^{-\lambda}}{k!} \qquad \text{(II.6)}$$

II.1.3.4. Bruit Gamma

Ce type de bruit peut être obtenu par un filtrage passe-bas des images basées par laser. [12][13]

$$k = \frac{a(b-1)^{b-1}}{(b-1)!} e^{-(b-1)} \tag{II.7}$$

$$P(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{pour } z \geq 0 \\ 0 & \text{pour } z < 0 \end{cases} \tag{II.8}$$

Où $\mu = \frac{a}{b}$ est ka moyenne et la variance est $\sigma^2 = \frac{b}{a^2}$

II.2. Filtrage

Le but principal du filtrage est amélioré son apparence. Cela se fait par :

- Améliorer la qualité visuelle d'une image en atténuant et/ou supprimant le bruit,
- Extraire des attributs d'une image (ex : contours),
- Modifier les valeurs des pixels de l'image,

En pratique, il s'agit de créer une nouvelle image en utilisant les valeurs de pixels de l'image d'origine. Les catégories filtrées n'incluent pas toutes les transformations de l'image d'origine telles que le zoom, le découpage, les projections, …

Dans le **filtrage global**, l'intensité de chaque pixel de la nouvelle image est calculée en prenant en compte la totalité des pixels de l'image de départ. Dans cette catégorie on trouve, par exemple, les opérations sur les histogrammes ou les opérations qui nécessitent de passer dans l'espace de Fourier. [14]

Dans **le filtrage local**, l'intensité de chaque pixel de la nouvelle image est calculée en prenant en compte seulement un voisinage du pixel correspondant dans l'image d'origine. Il est d'usage de choisir un voisinage carré et symétrique autour du pixel considéré. Pour le filtrage local on distingue deux types : filtrage linéaire et filtrage non linéaire. [14]

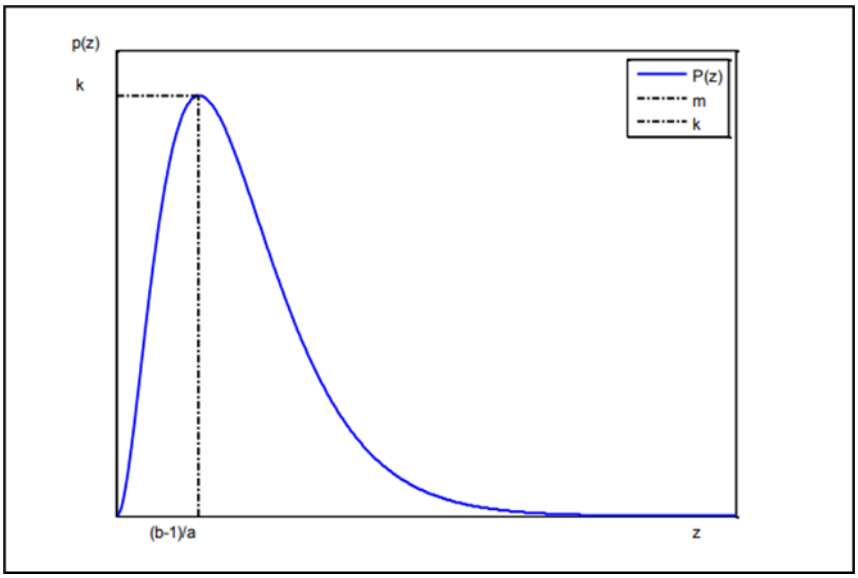


Figure II.4 : Distribution de bruit Gamma

II.2.1. **Filtres de convolution**

Plusieurs opérations de filtrages d'images utilisent la convolution. Cette dernière est caractérisée par un **masque** qui est une matrice de coefficients. La convolution est le processus consistant à ajouter chaque élément de l'image à ses voisins immédiats, pondéré par les éléments du noyau ou **masque**.

Par exemple, si nous avons deux matrices 3×3, la première étant le noyau et la seconde une partie de l'image, la convolution est le processus consistant à retourner les colonnes et les lignes du noyau puis de multiplier localement les valeurs ayant la même position, puis sommer le tout. L'élément aux coordonnées [2, 2] (l'élément central) de l'image de sortie devrait être pondéré par la combinaison de toutes les entrées de la matrice de l'image, avec les poids donnés par le noyau comme suit :

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

Les autres entrées devraient être pondérées de manière similaire, appliquée à tous les pixels de l'image. **Le noyau** ou **le masque** va donc agir sur chacun des pixels, c'est-à-dire sur chacun des éléments de la matrice "image".

Dans la figure II.5, l'image est représentée par la matrice [i] composée de n*m éléments. Le masque est quant à lui composé de la matrice carrée [k] de 3×3 éléments.

Appliquer un filtre de convolution consiste à multiplier chacun des pixels de la matrice [i] par le masque [K]. Pour calculer la valeur d'un pixel I(x, y) de la matrice image, on multiplie sa valeur par celle du pixel central du masque k(2, 2) et on additionne ensuite la valeur des produits des pixels adjacents. Il reste ensuite qu'à diviser le résultat par le nombre d'éléments du masque, cette dernière opération n'appartient pas au produit de convolution proprement dit, mais elle est nécessaire pour maintenir la dynamique de l'image (différence entre le niveau du pixel le plus élevé et le plus faible) ainsi que sa linéarité.

I(1,1)	I(1,2)	.						
I(2,1)	.							
.		.						
.			I(x,y)					
								I(n ,m)

*

K(1,1)	K(1,1)	K(1,1)
K(2,1)	K(2,2)	K(2,3)
K(3,1)	K(3,2)	K(3,3)

Figure II.5 : Le produit de convolution entre l'image I et le masque K

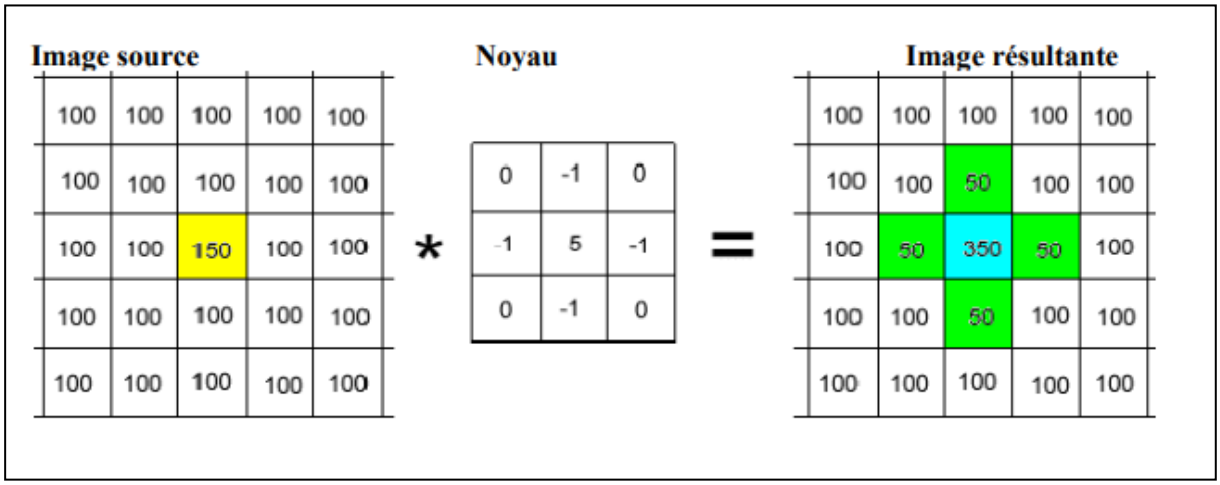


Figure II.6 : Exemple de filtrage d'une image par filtre de convolution

II.2.2. **Quelques méthodes de filtrage**

Dans la littérature, différentes méthodes de filtrage ont été développées suivant le type et l'intensité du bruit. Les premières et les plus simples de ces méthodes sont basées sur le filtrage linéaire stationnaire (invariant par translation), mais les limitations de ces technique (en particulier leur mauvaise conservation des transitions) a conduit au développement des filtres 'non linéaire'.

II.2.2.1. Filtrage linéaire

Le filtrage linéaire est la convolution d'une image $I(x,y)$ avec une fonction $f(x,y)$ qui s'appelle réponse impulsionnelles du filtre. Dans le cas continu, l'image filtrée est donnée par :

$$I(x,u) = (f * I)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x',y').I(x-x',y-y')dx'dy' \qquad (II.9)$$

Dans le cas discret, les domaines de I et de f sont bornés.

Le domaine de I est $\left[-\frac{N}{2},+\frac{N}{2}\right]$ et le domaine de f est $\left[-\frac{K}{2},+\frac{K}{2}\right]$

On a nécessairement K ≤ N, N étant la taille de l'image. Dans le cas discret la convolution s'écrit par:

$$If(x,u) = (f * I)(x,y) = \sum_{i'=-\frac{K}{2}}^{+\frac{K}{2}} \sum_{j'=-\frac{N}{2}}^{+\frac{N}{2}} f(i - i', j - j').I(i', j') \quad \text{(II.10)}$$

Le filtrage linéaire consiste donc à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des points voisins, les coefficients de cette combinaison linéaire sont définis par la réponse impulsionnelle du filtre. [13]

Les filtres linéaires les plus connus sont les filtres passe-bas et passe-haut.

a. Filtres Passe-bas (Lissage)

Les filtres « passe bas » agissent en sens inverse des filtres « passe haut » et le résultat est, un adoucissement des détails, ainsi qu'une réduction du bruit granuleux.

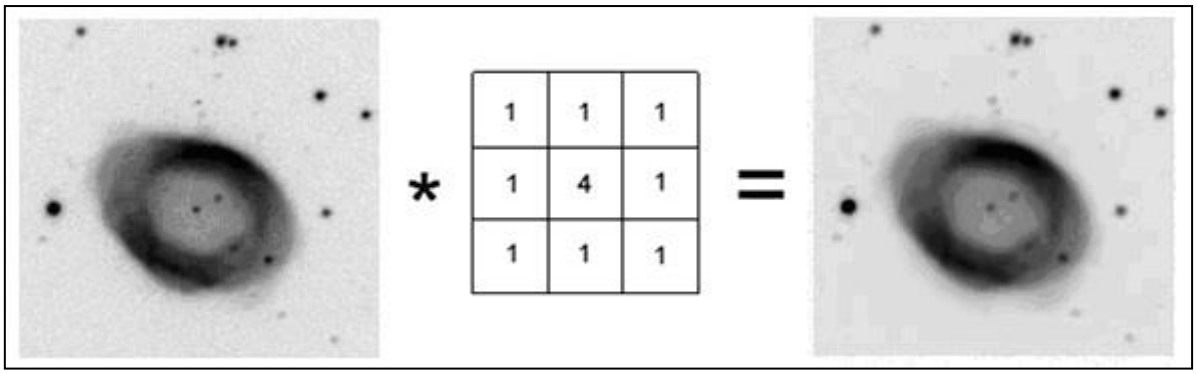


Figure II.7 : Exemple du filtrage passe-bas appliqué à une image [14]

b. Filtres Passe-haut (Accentuation)

Un filtre « passe haut » favorise les hautes fréquences spatiales, comme les détails, et de ce fait, il améliore le contraste. Un filtre « passe haut » est caractérisé par un noyau comportant des valeurs négatives autour du pixel central, comme dans l'exemple *de la figure II.8.*

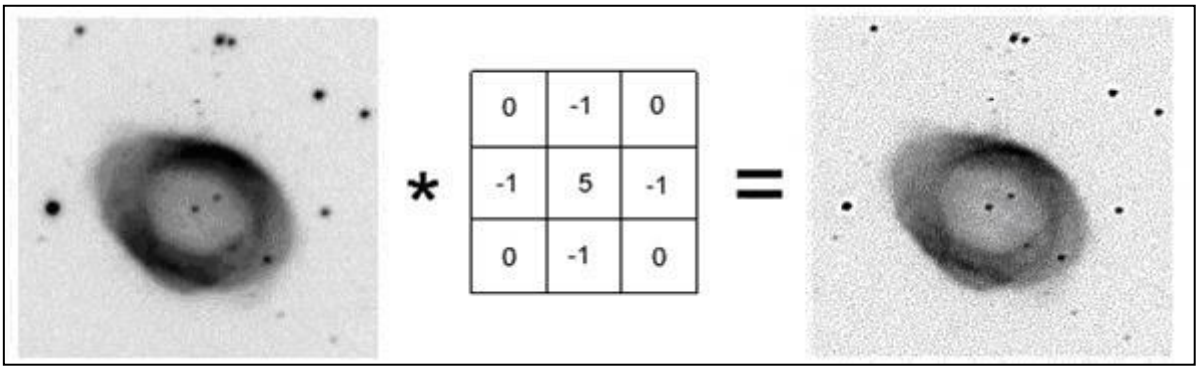


Figure II.8 : Exemple du filtrage passe-haut appliqué à une image [14]

II.2.2.2. Filtrage non-linéaire

Ces opérateurs ont été développés pour pallier aux insuffisances des Filtres linéaires et principalement la mauvaise conservation des contours. Ils ont le défaut d'infliger des déformations irréversibles à l'image résultante. La théorie des filtres non-linéaires est que chacun fondé sur des bases mathématiques ou empiriques différentes, ils permettent donc de supprimer totalement le bruit dans une image, tel que le filtre Médian.

Les filtres non-linéaires, ne sont pas des algorithmes prédéfinis puisqu'ils ne reposent pas sur une théorie mais ils sont plus variés. Souvent, ils sont plus coûteux en temps de calculs.

Un de leur avantage, est de permettre de s'adapter à la nature du point considéré (pixel utile ou pixel de bruit)

Deux aspects du lissage sont concernés par le filtrage non linéaire :

- Le bruit impulsionnel : les filtres linéaires éliminent mal les valeurs aberrantes,
- L'intégrité des frontières : on souhaiterait éliminer le bruit sans le rendre flous les frontières des objets.

II.2.3. **Quelques exemples de filtres**

II.2.3.1. Filtre gaussien

Le filtre gaussien est un opérateur de lissage, il donne une meilleure réduction du bruit que le filtre moyenne (Figure II.9), avec le filtre gaussien les contours et les détails fins sont mieux conservés. La fonction gaussienne à une forme caractéristique de courbe en cloche. [15]

Elle est aussi souvent utilisée dans les distributions statistiques, elle est définie par la fonction G(x) suivant :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-m)^2}{2\sigma^2}} \qquad \text{(II.11)}$$

Dont la représentation graphique de la fonction gaussienne est donnée à la figure II.9.

Dans le traitement d'images on traite des données à deux dimensions (x, y), on introduit donc une fonction gaussienne à deux dimensions G(x, y) :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})} \quad \text{(II.12)}$$

Où : σ : est l'écart type et G(x, y) : Coefficients de filtre gaussien obtenir par la fonction gaussienne G aux point (x,y).

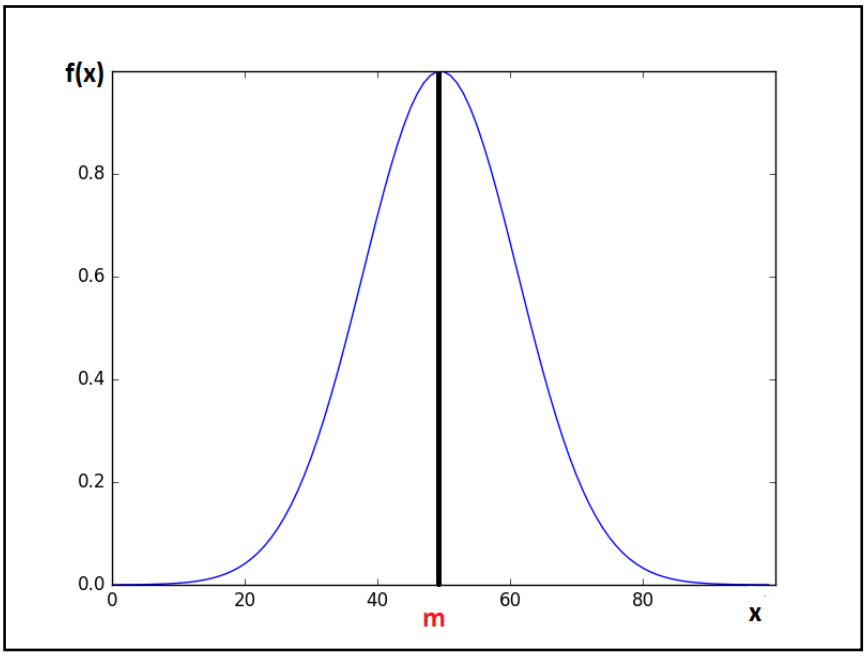


Figure II.9 : Les représentations graphiques d'une fonction gaussienne pour σ=1

Exemple : si σ = 0.8 ; on a le filtre 3*3 suivant:

G(-1,-1)	G(0,-1)	G(-1,-1)
G(-1,0)	G(0,0)	G(1,0)
G(-1,1)	G(0,1)	G(1,1)

$\approx \quad \frac{1}{16} *$

1	2	1
2	4	2
1	2	1

Et pour σ = 1 ; on a le filtre 5*5 suivant :

$$\approx \frac{1}{300}$$

1	4	6	4	1
4	18	30	18	4
6	30	48	30	6
4	18	30	18	4
1	4	6	4	1

En général un filtre gaussien avec σ < 1 est utilisé pour réduire le bruit, et si σ > 1 c'est dans le but de fabriquer une image qu'on va utiliser pour faire un « masque flou ».Donc plus σ est grand, plus le flou applique à l'image sera marqué. [15]

Les inconvénients du filtre gaussien sont :

- Gros lissage, filtre très large, gros problèmes de bord, et gros coût de calcul.
- Mélange de structures voisines.
- Perte de contraste.
- Délocalisation des bords.

II.2.3.2. Filtre moyenneur

La valeur du pixel central est remplacée par la moyenne des valeurs des pixels voisins dans le but de modifier les niveaux de gris trop différents de leurs voisins. En appliquant l'équation mathématique suivant :

$$I'(x,y) = \left(\frac{1}{d^2}\right) \sum_{(m,n)\in v} h(m,n) I(x-m, y-n) \qquad \text{(II.13)}$$

Où :

- **I (x, y)** : L'intensité du pixel voisin avec les coordonnées spatial **(x, y)** de l'image d'origine.
- **I′(x, y)** : l'intensité du pixel avec les coordonnées spatial **(x, y)** de l'image filtrée.

- **v** : Le voisinage.
- **h**: Masque de convolution avec les cordonnées **(m, n)** .
- **d**: taille du masque de Convolution.

La figure ci-dessous présente un exemple de filtre moyenneur avec un masque de convolution **h** de taille **3*3** :

$$I'(x,y) = \left(\frac{1}{3^2}\right)[1\times 51 + 1\times 21 + 1\times 14 + 1\times 9 + 1\times 9 + 1\times 19 + 1\times 19 + 1\times 22 + 1\times 21 = 7] = 21 \quad \text{(II.14)}$$

Où :

h=1/9*

1	1	1
1	1	1
1	1	1

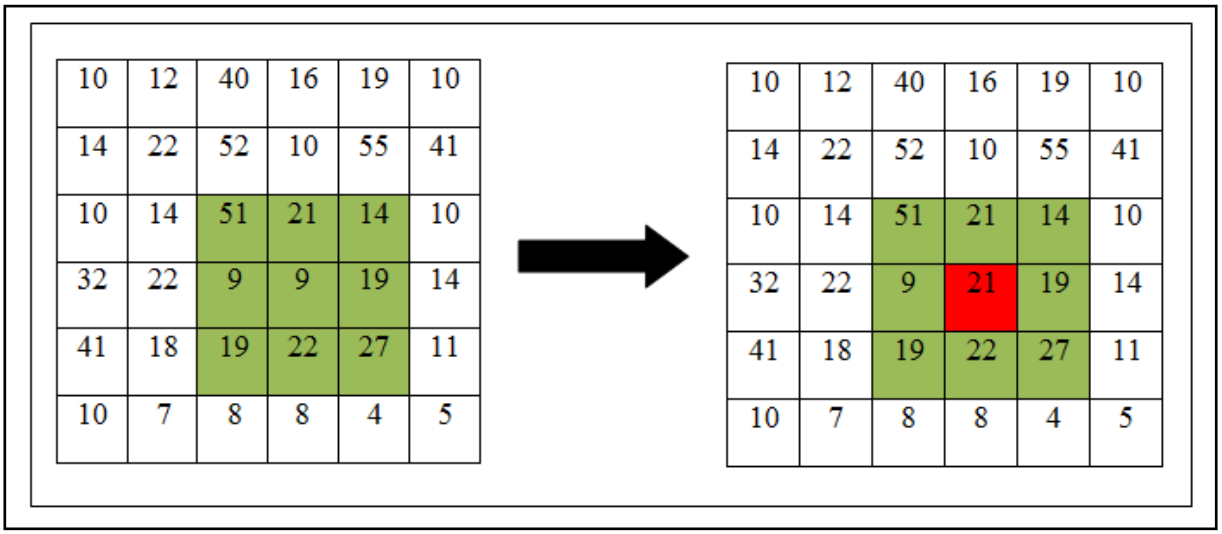


Figure II.10 : Illustration du principe du filtre moyenneur avec un noyau de taille 3*3

Dans l'exemple ci-dessus on voit un masque 3x3. Utiliser un noyau plus gros comme un 5x5 ou plus grand encore se révèle très lourd et peut créer une apparence artificielle. Les effets de ce filtre varient avec la taille du noyau, plus les dimensions du noyau seront importantes, plus le bruit sera éliminé, mais en contrepartie, les détails fins seront eux-aussi effacés et les contours étalés. C'est ce que nous illustrons ci-dessous (Figure II.11) pour des masques de lissage n*n, pour n croissante :

(a) (b) (c) (d)

Figure II.11 : Exemple de filtrage moyenneur avec différentes tailles de voisinage [16]

(a) Image originale, (b) Moyenneur 3*3, (c) Moyenneur 7*7 ; (d) Moyenneur 15*15.

En particulier, certains détails de l'image peuvent être éliminés (comme la texture des cheveux et de la moustache). Aussi dans certaines circonstances, un léger lissage (avec un masque 3*3 par exemple) peut être utilisé pour améliorer l'aspect esthétique d'une image.

Les inconvénients évidents de ce filtre de moyenneur sont les suivants:

- Un pixel isolé avec un niveau de gris anormal pour son voisinage va perturber les valeurs moyennes des pixels de son voisinage.
- Sur une frontière de régions le filtre va estomper le contour et le rendre flou, ce qui est gênant en visualisation bien sûr mais éventuellement aussi pour un traitement ultérieur qui nécessiterait des frontières nettes.
- Plus le filtre grossit plus le lissage devient important et plus le flou s'accentue.

II.2.3.3. Filtre médian

Le principe de ce filtre est de trier les pixels voisins par ordre croissant et de prendre la médiane (figure II.12) :

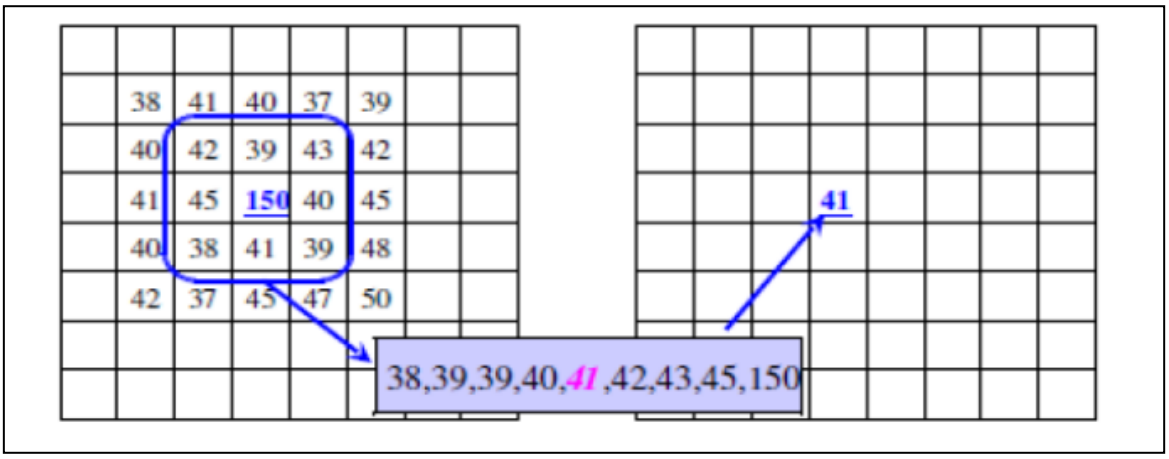


Figure II.12 : Principe de filtre Médian

- Les pixels voisins sont triés suivant un ordre croissant de 38 vers 150, puis la valeur médiane de la série numérique obtenue (41) devra remplacer le point central.

Le filtre médian sert à diminuer plusieurs types de bruits. On remarque que ce filtre est plus efficace dans le cas d'un bruit de type « sel et poivre ». En outre, ce filtre est connu par sa préservation de contour, mais affecte les angles et les détails fins. [13]

Intérêt du filtre médian :

- Un pixel non représentatif dans le voisinage affectera peu la valeur médiane.
- Elimine les bruits de type sel & poivre (Salt-and-Pepper).
- La valeur médiane choisie étant le niveau de gris d'un des pixels considérés, on ne crée pas alors de nouveaux niveaux de gris dans l'image. Ainsi lorsque le filtre passe sur un contour très marqué il le préservera mieux.

Inconvénients du filtre médian :

- Coûteux et nécessite un tri avant l'opération du filtrage.

II.2.3.4. Filtre maximum et minimum

a. Filtre maximum

On applique le même traitement que celui du filtre médian mais la valeur de pixel du centre va être substituée par le maximum [16].

Le filtre Max est utile pour éliminer le "poivre". Sur l'exemple ci-dessous le masque est un 3*3=9 éléments.les neufs éléments extraits de l'image sont ensuite triés dans l'ordre croissant la valeur max d'une série est définie (facilement repérable à cause du tri).

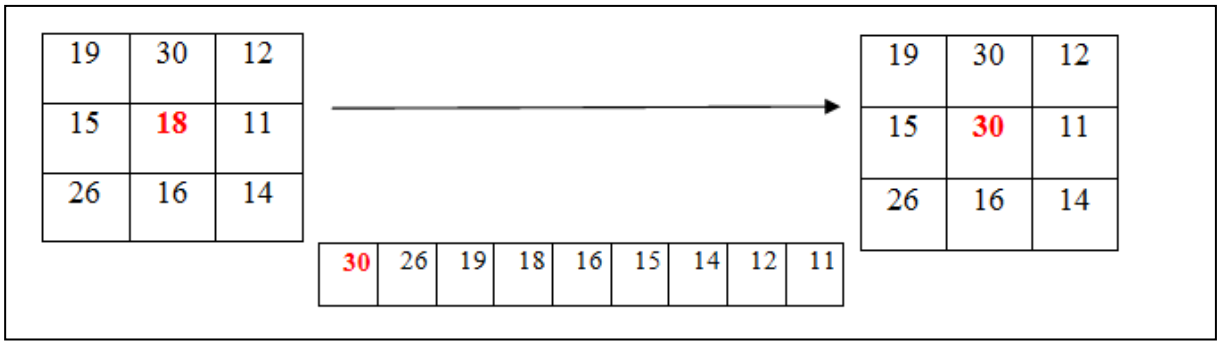


Figure II.13 : Le principe du filtre maximum avec une matrice de taille 3*3

b. Filtre minimum

On applique le même traitement que celui du filtre maximum mais la valeur de pixel du centre va être substituée par le minimum. [16]

Il est utile pour éliminer le "sel".

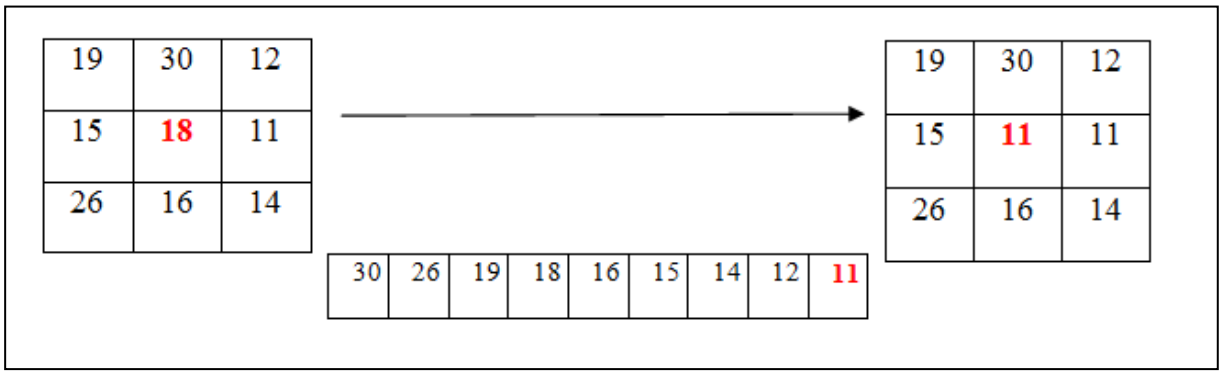


Figure II.14 : Le principe du filtre minimum avec une matrice de taille 3*3

II.2.3.5. Filtre Laplacien

Le filtre Laplacien est un produit de convolution particulier utilisé pour mettre en valeur les détails qui ont une variation rapide de luminosité. Il est souvent utilisé en amélioration d'images pour accentuer l'effet de contour. Le Laplacien est donc idéal pour rendre visible les contours des objets. [15]

Le Laplacien possède trois noyaux typiques de taille 3x3 :

Laplacien discret - 4

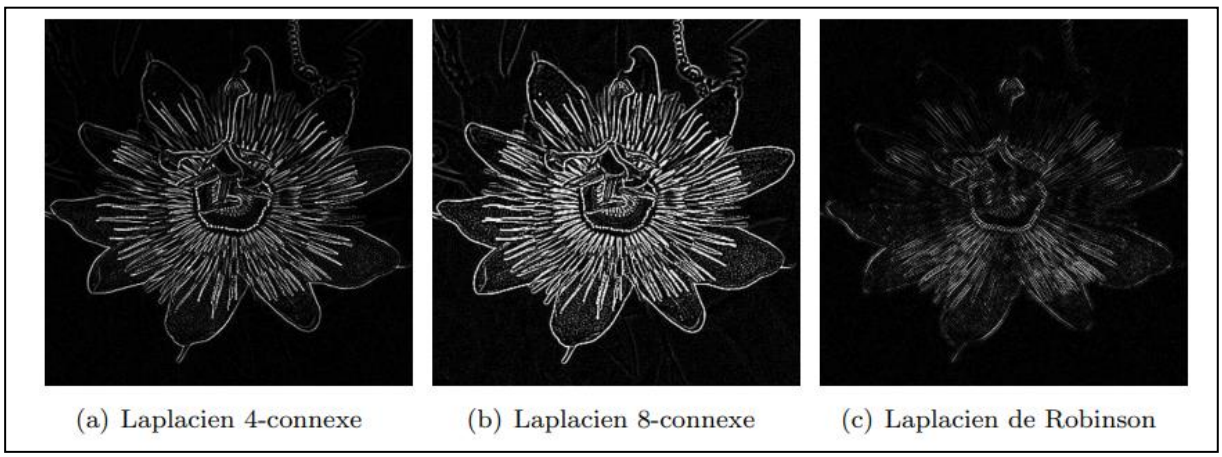
0	1	0
1	-4	1
0	1	0

Laplacien discret – 8

1	1	1
1	-8	1
1	1	1

Laplacien de Robinson

1	-2	1
-2	4	-2
1	-2	1



(a) Laplacien 4-connexe (b) Laplacien 8-connexe (c) Laplacien de Robinson

Figure II.15 : Filtre Laplacien [15]

Un détail est à noter : la somme de tous éléments du noyau d'un filtre Laplacien est toujours nulle, ce qui implique que ce filtre n'est pas un filtre linéaire.

II.2.3.6. Filtres morphologique

Toutes les transformations morphologiques sont définies à l'aide d'un élément structurant et un opérateur. L'élément structurant est un ensemble de pixels qui possèdent les caractéristiques suivantes :

- Une forme géométrique connue (voir la figure II.16).
- Un pixel central noir (voir la figure II.16).
- Un ensemble de pixels voisins au pixel central en blanc (voir la figure II.16).

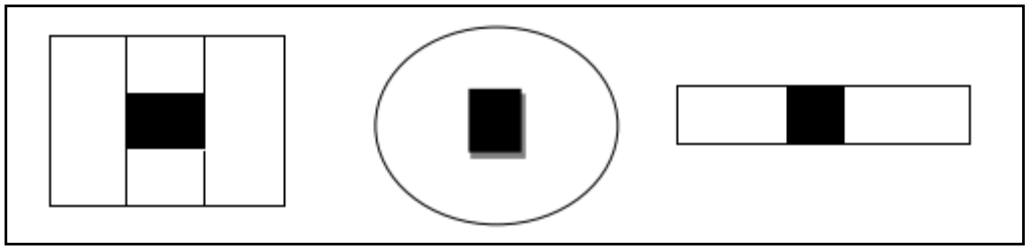


Figure II.16 : Éléments structurants [5]

Les deux principales opérations morphologiques sont : La *dilatation* et l'*érosion*, on peut trouver d'autres opérations comme :

- L'ouverture : c'est une opération qui consiste à éliminer des éléments fins et modifie les contours.
- La fermeture : c'est une dilatation qui permet de remplir les petits trous et de lisser les contours.

b. Dilatation et Erosion

❖ La Dilatation

Une dilatation consiste à déplacer l'élément structurant sur chaque pixel de l'image, et à regarder si l'élément structurant « touche » (ou plus formellement intersecte) la structure d'intérêt. Le résultat est une structure qui plus grosse que la structure d'origine (voir la figure II.17). En fonction de la taille de l'élément structurant, certaines particules peuvent se trouver connectées, et certains trous disparaître. [17]

❖ L'érosion

L'érosion est l'opération inverse, qui est définie comme une dilatation du complémentaire de la structure. Elle consiste à chercher tous les pixels pour lesquels l'élément structurant centré sur ce pixel touche l'extérieur de la structure. Le résultat est une structure rognée (voir la figure II.18). On observe la disparition des particules plus petites que l'élément structurant utilisé, et la séparation éventuelle des grosses particules. [17]

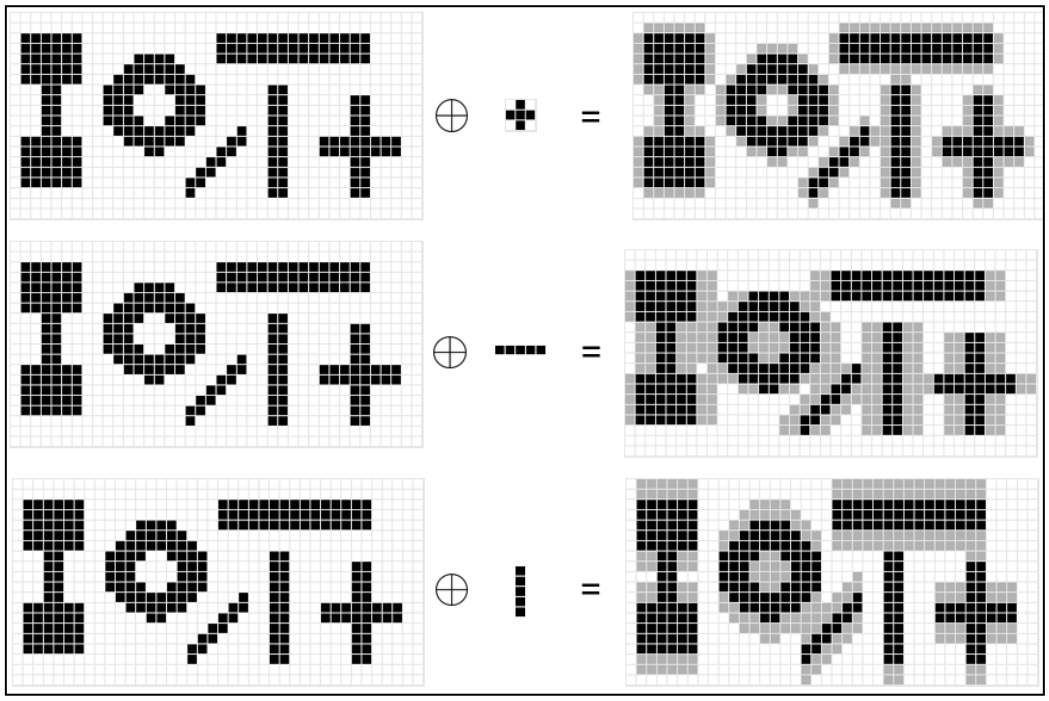


Figure II.17 : Exemple de dilatation sur des images binaires [18]

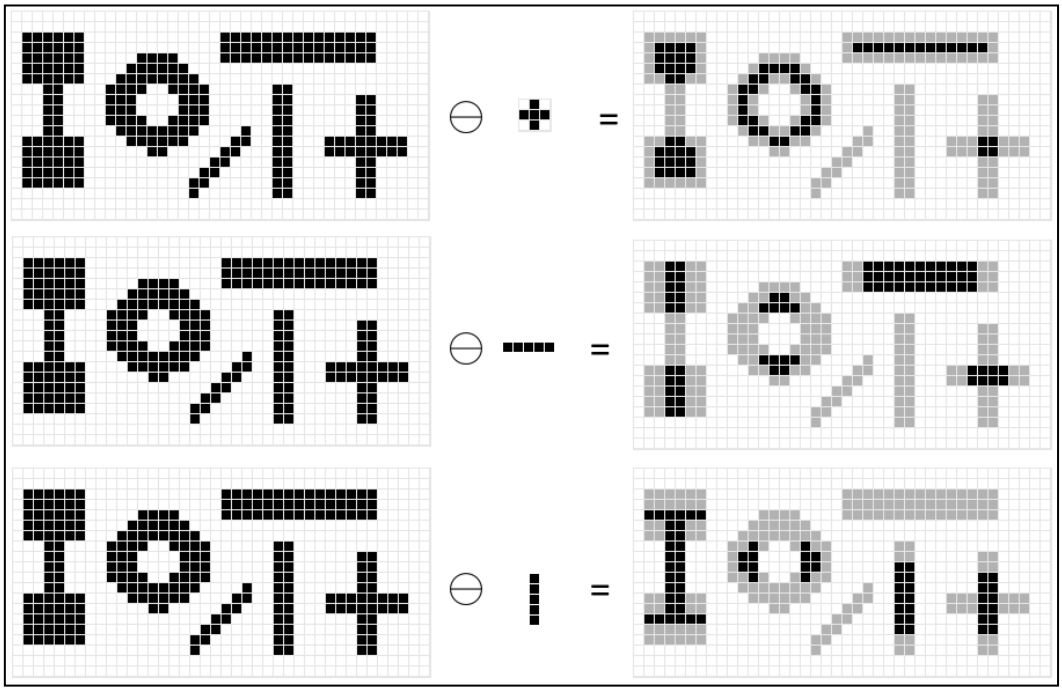


Figure II.18 : Exemple d'érosion sur des images binaire [18]

c. Ouverture et Fermeture

Ces deux opérations (ouverture et fermeture) sont définies à partir de la dilatation et de l'érosion.

❖ L'ouverture

C'est une érosion suivie d'une dilatation, On dit que l'ouverture rase les «pics » de l'histogramme sans modifier les « vallées ».

L'ouverture a pour effets de :

- faire disparaître les petites particules (dont la taille est inférieure à celle de l'élément structurant).
- séparer les grosses particules aux endroits où elles sont plus fines,

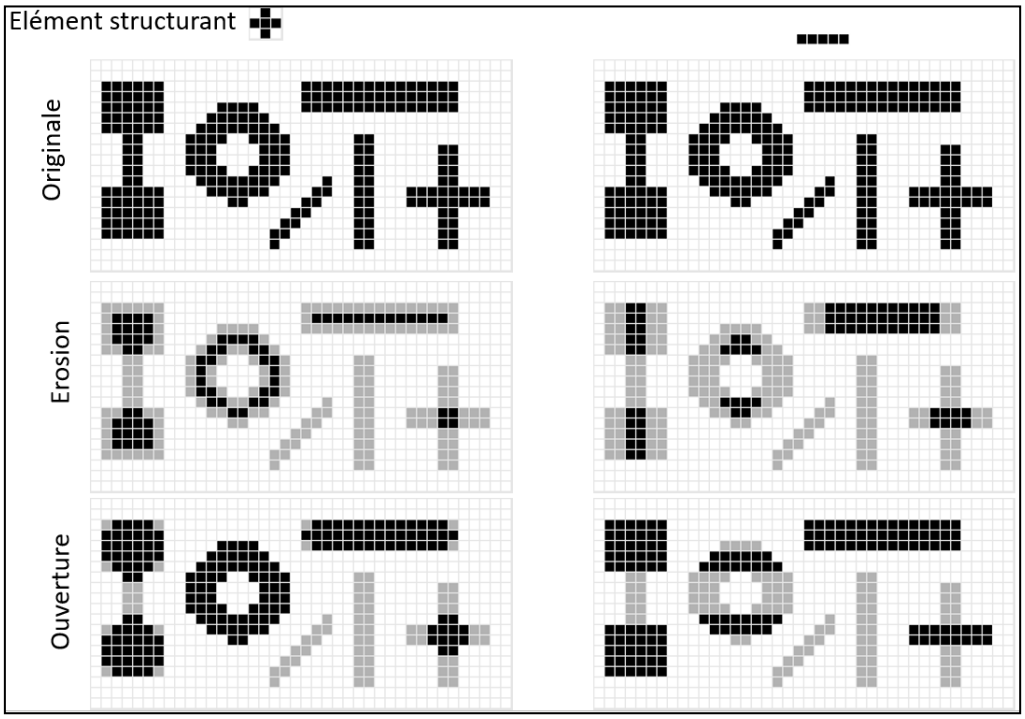


Figure II.19 : Exemples d'ouvertures avec un élément structurant en croix (colonne de gauche) et horizontal (colonne de droite) [18]

- La première ligne montre l'image de départ.
- La deuxième ligne montre le résultat de l'érosion.

- La dernière ligne montre le résultat final de l'ouverture (dilatation des images de la ligne précédente).
- L'opération d'ouverture a supprimé les éléments dans lequel il ne pouvait pas tenir entièrement.

❖ La Fermeture

C'est une dilatation suivie d'une érosion. On dira que la Fermeture comble les « vallées » sans modifier les «pics ».

La fermeture a pour effets de :

- Faire disparaître les trous de petite taille dans les structures.
- Connecter les structures proches.

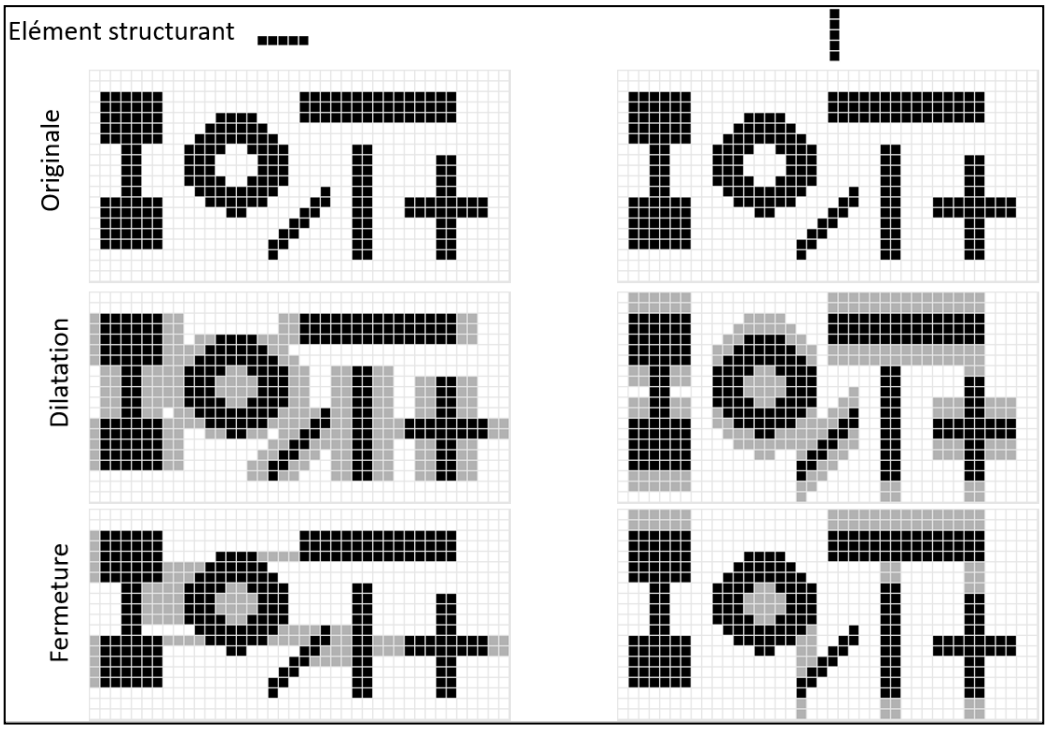


Figure II.20 : Exemples de fermetures avec un élément structurant horizontal (colonne de gauche) et vertical (colonne de droite) [18]

- La première ligne montre l'image de départ.
- La deuxième ligne montre le résultat de la dilatation.

- La dernière ligne montre le résultat final de la fermeture (érosion des images de la ligne précédente).
- L'opération de fermeture a rempli les espaces vides dans lequel il ne pouvait pas tenir entièrement.

II.2.3.7. Filtre bilatéral

Un filtre bilatéral est un filtre de lissage non linéaire , préservant les contours et réduisant le bruit pour les images . Il remplace l'intensité de chaque pixel par une moyenne pondérée des valeurs d'intensité des pixels voisins. Ce poids peut être basé sur une distribution gaussienne. Fondamentalement, les poids dépendent non seulement de la distance euclidienne des pixels, mais également des différences radiométriques (par exemple, les différences de portée, telles que l'intensité de la couleur, la distance en profondeur, etc.). Cela préserve les arêtes vives [19].

Le filtre bilatéral est défini comme :

$$I^{filtered}(x) = \frac{1}{W_p}\sum_{x_i \in \omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad \text{(II.15)}$$

et le terme de normalisation, , est défini comme W_P

$$W_p = \sum_{x_i \in \omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \quad \text{(II.16)}$$

Où :

- I^{filtered} est l'image filtrée ;
- I est l'image d'entrée d'origine à filtrer ;
- x sont les coordonnées du pixel courant à filtrer ;
- ω est la fenêtre centrée sur, ainsi qu'un autre pixel ; $xx_i \in \omega$
- f_r est le noyau de plage pour lisser les différences d'intensités (cette fonction peut être une fonction gaussienne)
- g_s est le noyau spatial (ou de domaine) pour lisser les différences de coordonnées (cette fonction peut être une fonction gaussienne).

Le poids est attribué en utilisant la proximité spatiale (en utilisant le noyau spatial) et la différence d'intensité (en utilisant le noyau de plage). Considérons un pixel situé à qui doit

être débruité dans l'image en utilisant ses pixels voisins et l'un de ses pixels voisins est situé à proximité. Ensuite, en supposant que la plage et les noyaux spatiaux sont des noyaux gaussiens , le poids attribué au pixel pour débruiter le pixel est donné par : $W_p\ g_s\ f_r\ (i,j)(k,l)(k,l)(i,j)$

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|I(i,j-I(k,l)\|^2}{2\sigma_r^2}\right) \quad \text{(II.17)}$$

Où

$\sigma_d$et σ_r sont des paramètres de lissage,$(i,j)(k,l)$ et I (i, j) et I (k, l) sont l'intensité des pixels et respectivement.

Après avoir calculé les poids, normalisez-les :

$$I_D(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)} \quad \text{(II.18)}$$

Où : $I_D(i,j)$ est l'intensité débruitée du pixel.

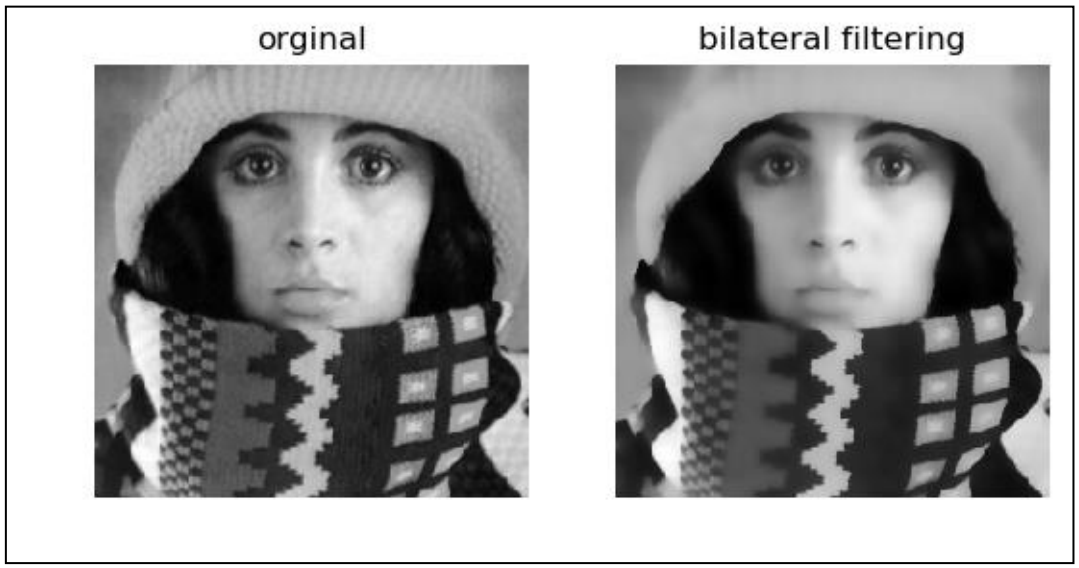


Figure II.21 : Le filtre bilatéral

II.2.3.8. Filtre de canny

Le filtre de Canny (ou détecteur de Canny) est utilisé en traitement d'images pour la détection des contours. L'algorithme a été conçu par John Canny en 1986 pour être optimal suivant trois critères clairement explicités [20]:

- Une bonne détection : faible taux d'erreur dans la signalisation des contours,
- Une bonne localisation : minimisation des distances entre les contours détectés et les contours réels,
- Clarté de la réponse : une seule réponse par contour et pas de faux positifs

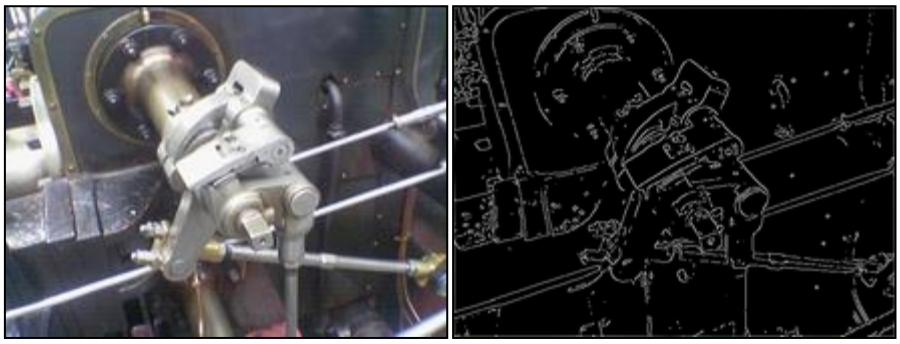


Figure II.22 : Le filtre de Canny

II.2.3.9. Filtre Gradient

Un gradient permet de visualiser les variations d'un phénomène, ainsi un dégradé de couleur peut s'appeler un gradient de couleur.

Mais parfois il est intéressant d'avoir de mettre en avant des détails qui changent selon une direction donnée, dans ce but on utilise des filtres de type gradient, qui sont en fait des dérivées partielles le long d'une direction particulière, en général l'un des deux axes cartésiens X ou Y de l'image.

Si les dérivées partielles sont supposées continues en un point quelconque de l'image, alors il est admis que la différentielle de la fonction Image en un point *M(x, y)* est une fonction linéaire qui est notée :

$$dz = f'_x(x,y).dx + f'_y(x,y).dy \qquad \text{(II.19)}$$

Écrit plus généralement sous cette forme :

$$dz = \frac{\partial f(x,y)}{\partial x}.dx + \frac{\partial f(x,y)}{\partial y}.dy \tag{II.20}$$

dz ou $\nabla f(x, y)$: est appelé Gradient et a pour amplitude:

$$\|\nabla f(x, y)\| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \tag{II.21}$$

Sa direction est donnée par :

$$\theta = \text{Arc tan}\frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}} \tag{II.22}$$

L'analyse du vecteur gradient permet de mettre en évidence un contour. En effet l'amplitude du gradient indique une plus ou moins forte discontinuité et sa direction est par définition normale au contour. Le gradient est un opérateur directionnel.

Localement à un point de l'image, les dérivées partielles font l'objet d'une approximation par une méthode de différentiation.

$\frac{\partial f(x,y)}{\partial x}$ est évaluépar $\Delta_x f(x, y) = f(x + 1, y) - f(x, y)$ (II.23)

$\frac{\partial f(x,y)}{\partial y}$ est évaluépar $\Delta_y f(x, y) = f(x, y + 1) - f(x, y)$ (II.24)

D'un point de vue géométrique, le gradient évalué sur un point indique la direction où la variation des niveaux de gris est la plus grande. Ce vecteur a une longueur très faible sur les points se trouvant à l'intérieur d'une région homogène.

Par contre, sur les points se trouvant à la limite entre deux régions, le gradient à une longueur importante, il est orienté vers la région la plus claire et se trouve perpendiculaire au contour.

Ainsi un filtre Gradient permet de mettre en évidence les variations de niveaux de gris suivant un axe variable, ce qui aura pour effet de mettre en évidence les fronts et de révéler les textures.

Pour une direction donnée, des filtres Gradient seront utilisés pour augmenter ou bien réduire les fronts sur cette direction, ce sont en fait des dérivées partielles le long d'une direction particulière, en général l'un des deux axes cartésiens X ou Y de l'image.

D'un point de vue pratique, les deux dérivées partielles suivant la largeur et la hauteur peuvent être approximées respectivement par des opérateurs de convolution. Plus le noyau de convolution est grand, plus larges sont les contours. [21]

a. Principe du gradient

❖ Gradient:

Soit f(x,y), alors :

$$\nabla f = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad \text{(II.25)}$$

❖ **Magnitude du gradient:**

$$mag(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \text{(II.26)}$$

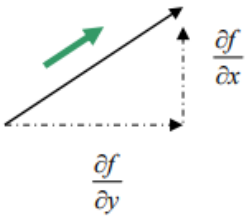
❖ **Approximation de la Magnitude:**

$$mag(\nabla f) \approx \left|\left(\frac{\partial f}{\partial x}\right)\right| + \left|\left(\frac{\partial f}{\partial y}\right)\right| \quad \text{(II.27)}$$

❖ **Direction du gradient :**

$$\theta = \text{Arc tan}\left[\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right] \quad \text{(II.28)}$$

Les approximations les plus simples des dérivées directionnelles se font par différences finies calculées par convolution avec des noyaux très simples.



b. Filtres de PREWITT, SOBEL, ROBERTS et KIRSCH:

Ces filtres, qui portent tous le nom de leurs inventeurs, sont tous conçus dans le même but : détecter avec la plus grande précision les contours naturels "cachés" dans un image.

A l'origine ils ont été développés dans le cadre des appareils de vision nocturne.

Le filtre de Sobel utilise par exemple deux noyaux 3x3, l'un pour l'axe horizontal (X) et l'autre pour l'axe vertical (Y) Chacun des noyaux est en fait un filtre gradient, qui sont tous les deux combinés pour créer l'image finale.

Ces masques sont généralement utilisés pour la détection des contours. Il s'agit des masques de Roberts, Sobel, Prewitt et Kirsh. [22]

❖ Masques de Sobel

On fait passer le masque de convolutions pour extraire les contours horizontaux puis verticaux. On additionne ensuite les deux images obtenues.

$$W_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} W_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{(II.29)}$$

W_x: direction horizontale , W_y: direction verticale

On remarque que Sobel est la combinaison d'un lissage dans un sens et d'un rehaussement dans l'autre:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times [1 \quad 0 \quad -1] \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \times [1 \quad 2 \quad 1] \quad \text{(II.30)}$$

❖ Masques de Roberts

$$W_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} W_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{(II.31)}$$

W_x: direction horizontale , W_y: direction verticale

❖ Masques de Prewitt

$$W_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} W_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{(II.32)}$$

W_x: direction horizontale , W_y: direction verticale

❖ Masques de Kirsh

$$W_x = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} W_y = \begin{bmatrix} -3 & -3 & 3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad \text{(II.33)}$$

W_x: direction horizontale , W_y: direction verticale

❖ Masques de Sobel pour 4 directions

$$W_{0^\circ} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} W_{45^\circ} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \text{(II.34)}$$

$$W_{90^\circ} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} W_{135^\circ} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

❖ Masques de Roberts pour 4 directions

$$W_{0^\circ} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} W_{45^\circ} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad \text{(II.35)}$$

$$W_{90^\circ} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} W_{135^\circ} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

❖ Masques de Prewitt pour 4 directions

$$W_{0^\circ} = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} W_{45^\circ} = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{(II.36)}$$

$$W_{90^\circ} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} W_{135^\circ} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

❖ Masques de Kirsh pour 4 directions

$$W_{0^\circ} = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} W_{45^\circ} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} \quad \text{(II.37)}$$

$$W_{90^\circ} = \begin{bmatrix} -3 & -3 & 3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} W_{135^\circ} = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$$

II.2.4. **Rapport signal à bruit en pic (PSNR)**

Lorsqu'on utilise les méthodes de traitement d'images, la mesure de la qualité de l'image est principe essentiel La mesure la plus couramment utilisée est le Peak Signal to Noise Ratio (PSNR), évaluant le rapport signal à bruit en pic et se mesure en décibel (dB). Pour les images en niveaux de gris le PSNR est calculé par :

$$PSNR = 10 \times \log_2 \frac{255^2}{MSE} \quad \text{(II.38)}$$

$$MSE = \frac{1}{N \times M} \times \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I(i,j) - \hat{I}(i,j))^2 \quad \text{(II.39)}$$

où : I : Image originale et Î : Image bruité.

$$PSNR = 10 \times \log_2 \frac{255^2 \times 3}{MSE(R) + MSE(G) + MSE(B)} \quad \text{(II.40)}$$

où MSE(·) représente les erreurs quadratiques moyennes dans les trois plans R, G et B.

II.3. Conclusion

Dans que ce chapitre, on a abordé les concepts et les principes fréquemment utilisés aux traitements d'images. Ces notions concernent le bruit et les filtres principalement utilisés pour l'amélioration des images entachées de bruit. Le traitement d'images améliore la qualité des images en réduisant le bruit et en améliorant les propriétés de l'image traitée pour la rapprocher d'image originale afin qu'elle puisse être utilisée dans d'autres applications. Le chapitre suivant discutera les résultats obtenus sur des images bruitées avec déférents types filtrages étudiés.

Chapitre III

Expérimentations et résultats

III.1. Introduction

Ce chapitre porte sur l'expérimentation de la méthode de filtrage décrite dans la section précédente. Tout d'abord, avant de définir l'implémentation de l'algorithme, il présente l'environnement matériel et la plate-forme utilisés pour ce travail et enfin, pour terminer, faire le test et voir les résultats de ces expérimentations.

III.2. Environnement de travail

III.2.1. Matériel utilisé

Notre travail s'est effectué sur un micro-ordinateur ayant les caractéristiques suivantes :

- Processeur : Dual Core Intel 2,9 GHz
- RAM : 4 Go
- Ecran : Samsung 60 Hz 19 pouces.
- Système d'exploitation : Windows 7 64 bits

III.2.2. Outils de développement

Nous avons travaillé dans ce chapitre pour créer un système de filtrage dans lequel l'algorithme est codé sera en ligne, en utilisantGoogle Colab (abréviation de Colaboratory) avec langage de programmation python. D'autre part, OpenCV (Open Source Computer Vision) est la bibliothèque utilisée contribuant à l'implémentation du projet.

III.2.2.1. Google Colab

Colab (abréviation de Colaboratory) est un produit proposé par Google Research qui permet chercheurs en apprentissage automatique pour travailler sur des projets dans le navigateur. Similaire à Google Docs, Il nous permet de partager des projets entre plusieurs personnes, et le meilleur de tous, qu'il offre un accès gratuit aux GPU afin que nous puissions former des modèles rapidement sans aucun enregistrement. [23]

Comme définition générale, Colab est un environnement de notebook gratuit qui s'exécute entièrement dans le cloud. Mieux encore, il ne nécessite aucune configuration et les blocs-notes que nous créons peuvent être modifiés simultanément par les membres de notre équipe, tout comme nous modifions des documents dans Google Docs. Colab prend en charge de

nombreuses bibliothèques de machine learning populaires qui peut être facilement chargées sur notre ordinateur.

❖ Les offres de Google Colab

En tant que programmeurs, nous pouvons effectuer les opérations suivantes à l'aide de Google Colab:

- Écrire et exécuter du code en Python.
- Documenter notre code qui prend en charge les équations mathématiques.
- Créer/Télécharger/Partager des blocs-notes.
- Importer/Enregistrer des blocs-notes depuis/vers Google Drive.
- Importer/publier des notebooks depuis GitHub.
- Importer des ensembles de données externes.
- Intégrer PyTorch, TensorFlow, Keras, OpenCV.
- Service Cloud gratuit avec GPU gratuit.

III.2.2.2. Langage de programmation

Python est le langage de programmation open source le plus couramment utilisé par les informaticiens. Ce langage se démarque dans les domaines de la gestion d'infrastructures, de l'analyse de données ou du développement de logiciels. En fait, dans ses qualités, Python permet aux développeurs de se concentrer sur ce qu'ils font, et non sur la façon dont ils le font. Il libère les développeurs des contraintes formelles des anciens langages qui tourmentaient leur époque. Par conséquent, développer du code en Python est plus rapide que dans d'autres langages.

Il est également disponible pour les débutants si nous passons du temps à nous lancer. De nombreux tutoriels sont également disponibles sur des sites dédiés. Les réponses aux questions peuvent toujours être trouvées sur les forums informatiques, car de nombreux professionnels l'utilisent.

❖ Bibliothèques en Python

Normalement, une bibliothèque est une collection de livres ou une pièce ou un lieu où de nombreux livres sont stockés pour être utilisés plus tard. De même, dans le monde de la programmation, une bibliothèque est une collection de codes précompilés qui peuvent être

utilisés ultérieurement dans un programme pour certaines opérations spécifiques bien définies. Outre les codes précompilés, une bibliothèque peut contenir de la documentation, des données de configuration, des modèles de message, des classes et des valeurs, etc. [24]

Une bibliothèque Python est une collection de modules associés. Il contient des paquets de code qui peuvent être utilisés à plusieurs reprises dans différents programmes. Cela rend la programmation Python plus simple et pratique pour le programmeur. Comme nous n'avons pas besoin d'écrire le même code encore et encore pour différents programmes. Les bibliothèques Python jouent un rôle très important dans les domaines de l'apprentissage automatique, de la science des données, de la visualisation des données, etc. [24]

❖ Bibliothèques utilisés

Matplotlib : cette bibliothèque est chargée de tracer des données numériques. Et c'est pourquoi il est utilisé dans l'analyse des données. C'est également une bibliothèque open-source et trace des figures hautes définition comme des camemberts, des histogrammes, des nuages de points, des graphiques, etc.

Pandas : Les pandas sont une librairie importante pour les data scientists. Il s'agit d'une bibliothèque d'apprentissage automatique open source qui fournit des structures de données flexibles de haut niveau et une variété d'outils d'analyse. Il facilite l'analyse des données, la manipulation des données et le nettoyage des données. Les pandas prennent en charge des opérations telles que le tri, la réindexation, l'itération, la concaténation, la conversion de données, les visualisations, les agrégations, etc.

Numpy : Le nom « Numpy » signifie « Numerical Python ». C'est la bibliothèque couramment utilisée. Il s'agit d'une bibliothèque d'apprentissage automatique populaire qui prend en charge de grandes matrices et des données multidimensionnelles. Il se compose de fonctions mathématiques intégrées pour des calculs faciles. Même des bibliothèques comme TensorFlow utilisent Numpy en interne pour effectuer plusieurs opérations sur les tenseurs. Array Interface est l'une des principales caractéristiques de cette bibliothèque.

openCV : une énorme bibliothèque open source pour la vision par ordinateur, l'apprentissage automatique et le traitement d'images. OpenCV prend en charge une grande variété de langages de programmation comme Python, C++, Java, etc. Il peut traiter des images et des vidéos pour identifier des objets, des visages ou même l'écriture manuscrite d'un

humain. Lorsqu'il est intégré à diverses bibliothèques, telles que Numpy qui est une bibliothèque hautement optimisée pour les opérations numériques, le nombre d'armes augmente dans votre Arsenal, c'est-à-dire que toutes les opérations que l'on peut effectuer dans Numpy peuvent être combinées avec OpenCV.

III.3. Mécanisme de travail

Il existe d'autres méthodes qui sont très largement utilisées dans le monde du filtrage d'images. Notre but n'étant pas d'écrire un catalogue des différentes techniques mais de comparer les performances de quelques filtres sur une image bruitée par différents bruit utilisant des critères employés en traitement d'image.

Nous avons divisé notre travail en 6 étapes :

- **Etape 01 :** Comment créer un projet dans Google Colab et expliquer l'interface.
- **Etape 02 :** Ajouter les bibliothèques utilisées dans le travail, et lier le travail à notre Google Drive.
- **Etape 03 :**Les images utilisées et certaines de ses caractéristiques.
- **Etape 04 :** Le bruitage sera appliqué sur l'image choisie. Nous appliquerons deux types de bruits : bruit sel et poivre et bruit gaussien.
- **Etape 05 :** Les images bruitées sera filtrées et afficher les résultats.
- **Etape 06 :** L'analyse des performances des filtres (Discussion).

III.4. Création de projet et l'explication de l'interface

Pour créer un nouveau projet dans Google Colab, nous ouvrons l'URL suivante dans notre navigateur : https://colab.research.google.com

Notre navigateur affichera l'écran suivant (En supposant que nous sommes connectés à notre Google Drive) :

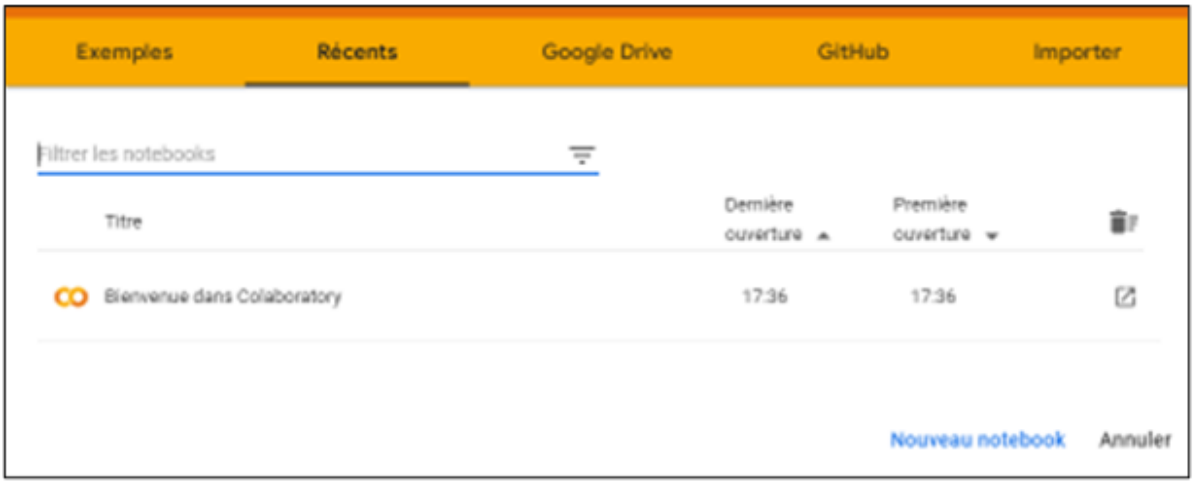


Figure III.1 : Création un projet dans Google Colab

Remarque : Nous devons avoir un compte Gmail pour utiliser cet outil. Si nous n'avons pas encore de compte Gmail, nous devrons en créer un.

Nous pouvons créer un nouveau fichier en cliquant sur NOUVEAU NOTEBOOK.

Ensuite, l'interface de travail apparaîtra.

- L'explication de l'interface

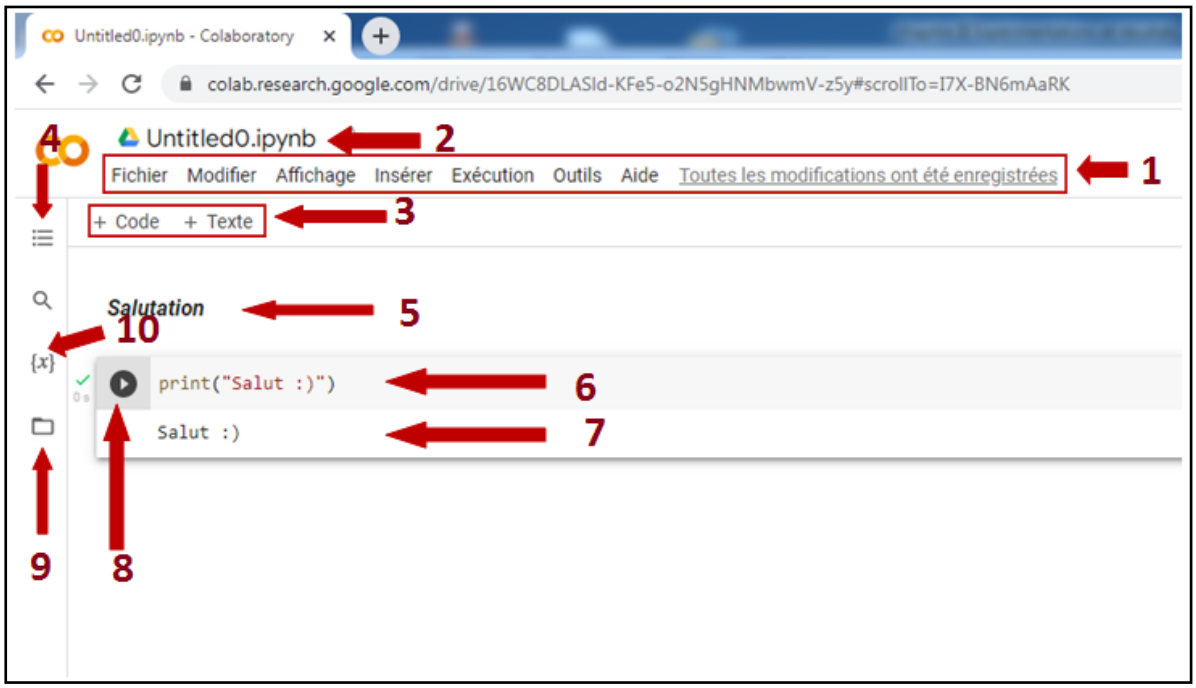


Figure III.2 : Partie gauche de l'interface

1. Barre de menus : comme dans toute autre application, cette barre de menus peut être utilisée pour manipuler le fichier entier ou ajouter de nouveaux fichiers. Permet de parcourir les différents onglets et de se familiariser avec les différentes options.

2. Nom de fichier : il s'agit du nom de votre fichier. Il peut être cliqué pour changer le nom. Il est préférable de ne pas faire le changement de l'extension (.ipynb) lors de la modification du nom car cela peut rendre le fichier qui ne s'ouvre pas.

3. Insérer une cellule de code ou texte : Il se compose de deux boutons à travers lesquels nous pouvons ajouter un code ou une cellule de texte sous la cellule précédente ou spécifique.

4. Table des matières : Ici, nous serons en mesure de créer et de parcourir différentes sections à l'intérieur de notre bloc-notes. Les sections nous permettent d'organiser les instructions logicielles et d'améliorer la possibilité de lire.

5. Cellule de texte : C'est ici que nous pouvons écrire un texte ou un titre expliquant la cellule de code.

6. Cellule de code : C'est ici que nous pouvons écrire notre code ou algorithme utilisé.

7. Sortie : il s'agit de la sortie de votre code, y compris les erreurs, qui seront affichées.

8. Exécuter la cellule : il s'agit du bouton d'exécution. Cliquer dessus exécutera tout code inséré dans la cellule à côté.

Remarque : Il y a deux indications lors de l'exécution du code

: Marque verte pour le correct processus d'exécution de code.

: Marque rouge pour le mauvais processus d'exécution de code.

9. Fichiers : ici, nous serons téléchargés des ensembles de données et d'autres fichiers depuis notre ordinateur et Google Drive.

10. Extraits de code : ici, nous pourrons trouver des extraits de code pré-écrits pour différentes fonctionnalités, telles que l'ajout de nouvelles bibliothèques ou le référencement d'une cellule à une autre.

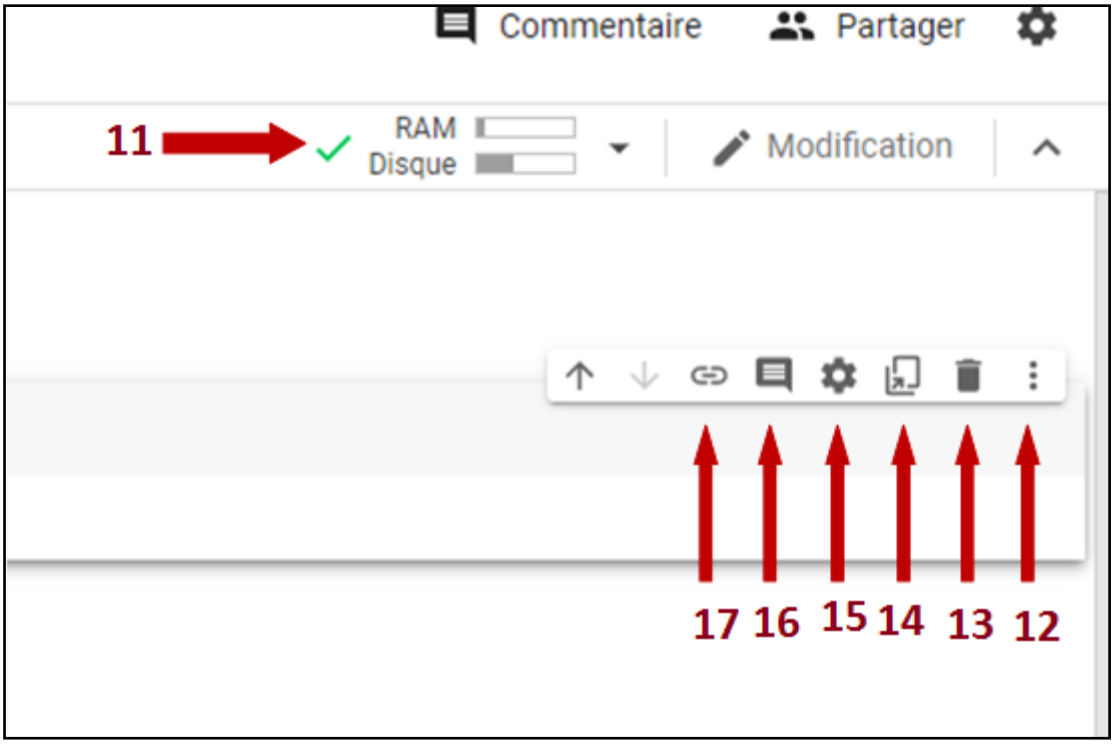


Figure III.3 : Partie droite de l'interface

11. RAM et disque : Tout le code que nous écrivons sera exécuté sur un ordinateur de Google et nous ne verrons que la sortie. Cela signifie que même si nous avons un ordinateur lent, le fonctionnement de grandes parties du code n'est pas un problème. Google spécialise une certaine quantité d'espace RAM et disque pour chaque utilisateur.

12. Plus d'options : Contient des options pour couper et copier une cellule ainsi que l'option pour ajouter un formulaire et masquer le code.

13. Supprimer la cellule : ce bouton supprimera la cellule sélectionnée.

14. Miroir la cellule : mettre en miroir la cellule dans un onglet.

15. Paramètres : ce bouton nous permettra de modifier le thème du notebook, le type de police, la taille, la largeur de l'indentation, etc.

16. Commentaire : Ce bouton nous permettra de créer un commentaire sur la cellule spécifiée. Ce sera un commentaire sur la cellule et non un commentaire dans la cellule.

17. Lien vers la cellule : ce bouton créera une URL qui sera liée à la cellule que nous avons sélectionnée.

III.5. Les bibliothèques utilisées

Avant d'écrire le code (algorithme) pour afficher l'image, nous devons d'abord mettre les bibliothèques utilisées dans notre travail (nous pouvons mettre chaque bibliothèque avec son code d'utilisation), puis nous lions notre travail à notre drive.

Parmi les avantages de Google Collab, il y a de nombreuses bibliothèques que nous n'avons pas besoin d'installer, nous écrivons simplement le code **« import + le nom de la bibliothèque »** et nous pouvons l'utiliser Et nous allons le raccourcir dans notre travail en utilisant le code **« as + n'importe nom »**

Exemple :

```
import numpy as np
import pandas as pd
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from skimage.io import imread , imshow
from skimage import exposure
from skimage.color import rgb2gray
import cv2
from google.colab.patches import cv2_imshow
from numpy import array
```

Figure III.4 : Les bibliothèques utilisées

Remarque : Le code **« from »** dans la figure pour sélectionner une bibliothèque spécifique et en extraire un code car nous n'avons besoin que de ce code, pas de toute la bibliothèque

Après avoir entré les codes des bibliothèques, nous connecterons le notebook à notre driveen utilisant le code suivant :

```
from google.colab import drive
drive.mount('/content/drive')
```

Lors de l'exécution, une fenêtre apparaîtra nous demandant d'autoriser le notebook à accéder aux fichiers du Google drive, nous appuyons sur **« Se connecter à Google Drive »**

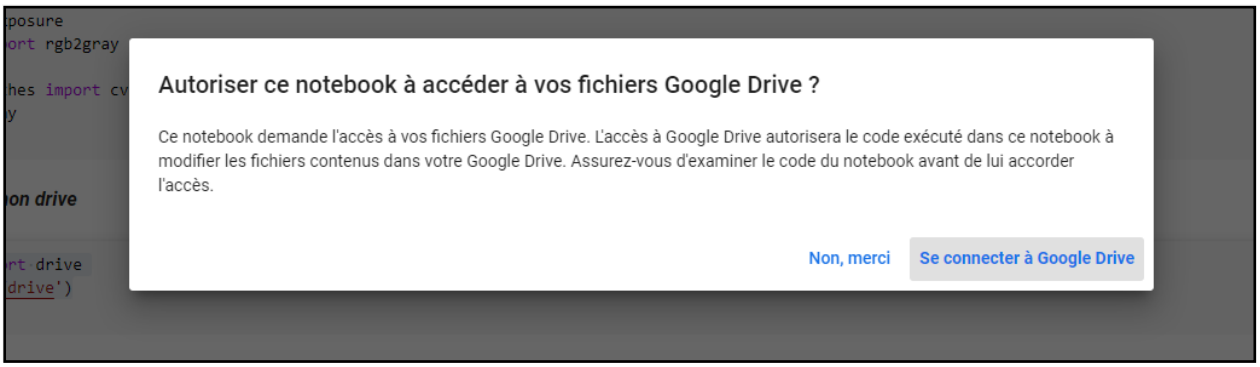


Figure III.5 : La fenêtre d'autorisation

Ensuite, une nouvelle fenêtre apparaîtra pour que nous sélectionnions notre compte Google, choisissez le compte et appuyez sur **« Autoriser »**

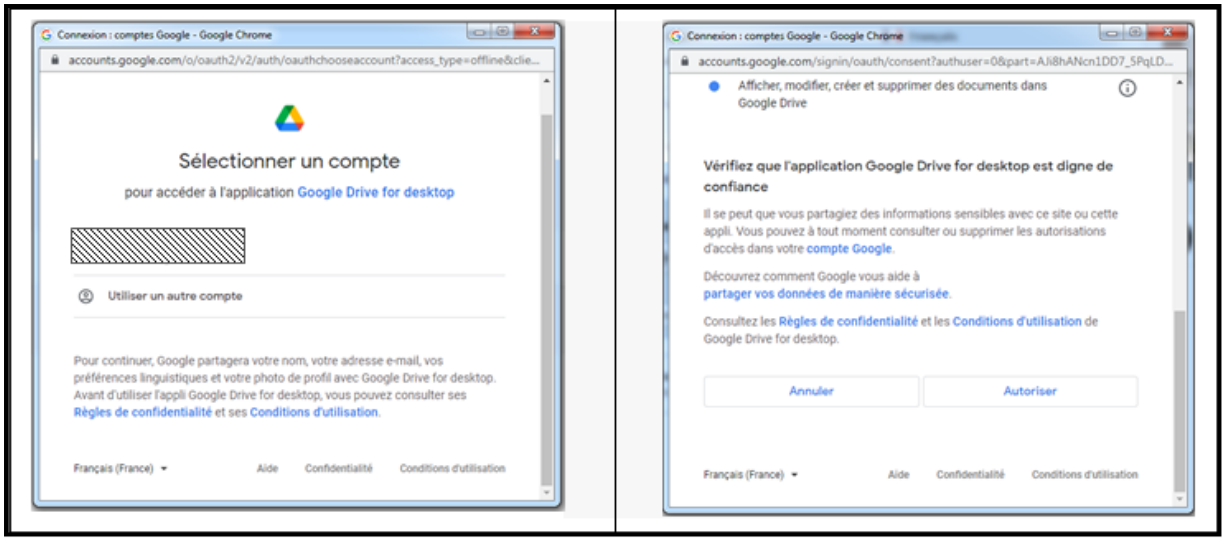


Figure III.6 : Les fenêtres de Google

Ainsi, nous avons lié le notebook à notre Google Drive.

III.6. Les images utilisées et certaines de ses caractéristiques

Les images que nous utiliserons dans le notebook (notre travail) doit être sur le bureau de l'ordinateur ou dans nos fichiers de Drive, et dans notre cas, nous le lirons à partir des fichiers du Drive en utilisant le code suivant :

```
image=cv2.imread('/content/drive/MyDrive/Lenna.png')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2_imshow(image)
```

Nous avons utilisé la fonction **imread()** de la bibliothèque cv2 pour lire l'image, Lorsque le fichier image est lu avec la fonction **imread()**, l'ordre des couleurs est BVR (bleu, vert, rouge). Nous devons convertir BVR et RVB. Nous utiliserons la fonction **cvt.Color()** et la fonction **imshow()** pour afficher l'image.

cv2.imread (chemin)

Paramètres:

- **chemin:** Une chaîne représentant le chemin de l'image à lire.

cv2.cvtColor(src, code)

Paramètres :

- **src :** C'est l'image dont l'espace colorimétrique doit être changé.
- **code :** C'est le code de conversion de l'espace colorimétrique.

cv2_imshow(image)

Paramètre :

- **image :** C'est l'image qui doit être affichée.

Remarque 1 : Le chemin des images utilisées sont :

Pour l'image de lena : ('/content/drive/MyDrive/Lenna.png')

Pour l'image de l'université : ('/content/drive/MyDrive/UnivBiskra.jpg')

Figure III.7 : L'image de Lena

Figure III.8 : L'image Université

Remarque 2 : Dans la fonction **imread**, il y a une case pour changer la couleur de l'image soit gris ou coloré, **cv2.imread (chemin, drapeau).**

- **drapeau :** Il spécifie la manière dont l'image doit être lue. Une valeur de 0 pour une image en niveaux de gris et 1 pour une image en couleur.

Exemple :

```
image=cv2.imread('/content/drive/MyDrive/Lenna.png',0)
cv2_imshow(image)
```

Figure III.9 : L'image de Lena grise

Après avoir montré l'image, nous insérerons des codes pour afficher certaines de ses caractéristiques.

La fonction **shape[]** donne la dimension et **dtype** le type de l'image comme suit :

```
print(image.dtype)
print(image.shape[0])
print(image.shape[1])
print(image.shape[2])
```

print pour afficher les résultats.

- **shape [0]** pour longueur de l'image.
- **shape [1]** pour largeur de l'image.
- **shape [2]** Pour savoir combien de canaux de couleur contient une image.

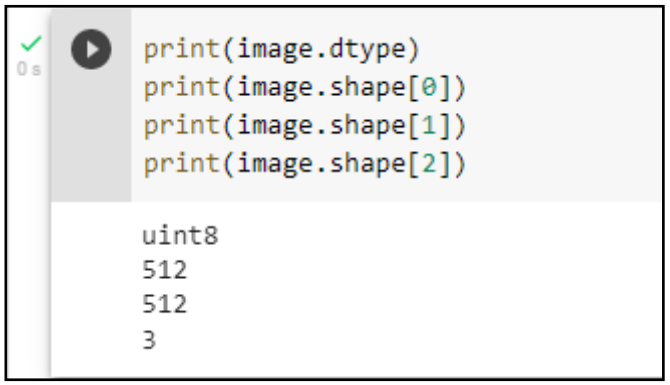


Figure III.10 : Dimension de l'image de Lena

Type de l'image **uint8** :

- **U :** Unsigned, Ce qui signifie : Toutes les valeurs sont positives.
- **INT :** Integers, Ce qui signifie : Toutes les valeurs sont des nombres entiers, 0, 1, 2,3...
- **8 :** Seulement 8 bits d'information, Ce qui signifie : la valeur max est 255 et la valeur min est 0.

Longueur et largeur de l'image est **512** pixels

Il se compose de 3 canaux de couleur (R : rouge, V : vert, B : bleu) Et nous montrerons ces canaux avec les deux fonctions **for** et **range (loop).**

```
rgb = ['Reds','Greens','Blues']
_, axes = plt.subplots(1, 3, figsize=(20,10), sharey = True)
for i in range(3):
  axes[i].imshow(image[:,:,i], cmap = rgb[i])
  axes[i].set_title(rgb[i], fontsize = 15)
```

La fonction **subplot()** de la bibliothèque **matplotlib** ajoute une sous-parcelle à une figure actuelle à la position de grille spécifiée.

plt.subplots (nrows, ncols, figsize,sharey)

Paramètres:

- **nrows, ncols :** Ces paramètres sont le nombre de lignes/colonnes de la grille des sous-parcelles.
- **Figsize :** L'attribut nous permet de spécifier la largeur et la hauteur d'une figure en pouces unitaires.
- **sharex, sharey :** Ces paramètres contrôlent le partage des propriétés entre les axes x (sharex) ou y (sharey).

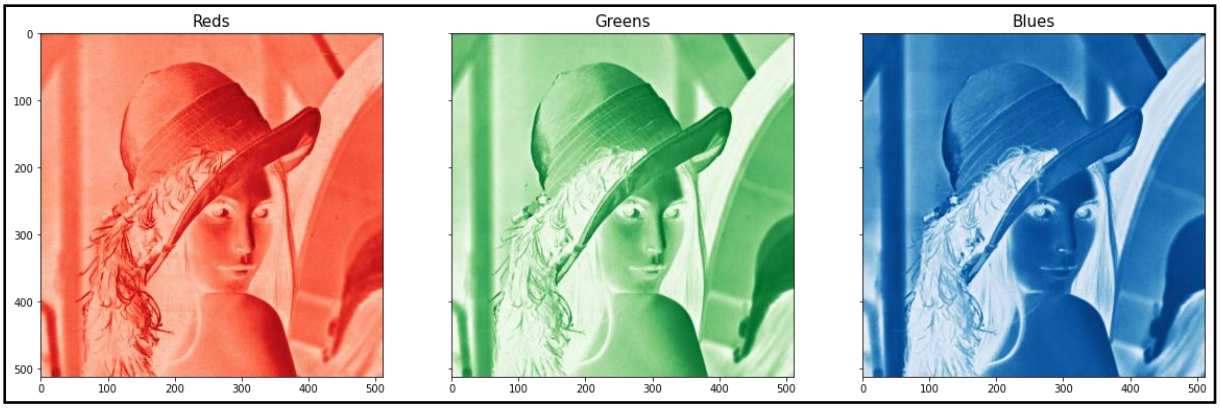


Figure III.11 : 3 canaux de couleur d'image Lena (RVB)

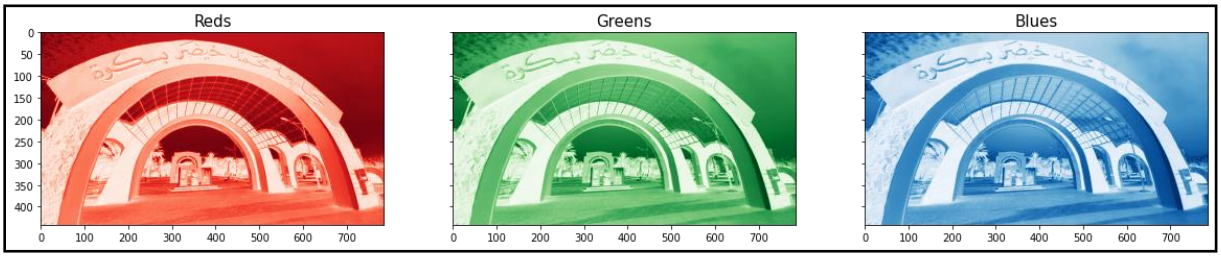


Figure III.12 : 3 canaux de couleur d'image Université (RVB)

Ensuite, nous représenterons histogramme RVB de l'image en utilisant la fonction **def** et **subplot** avec le code suivant :

```
def histColor(image): #Définition de la fonction
    _, axes = plt.subplots(ncols=2, figsize=(20, 10)) #la taille de la graph et il se compose de deu
x colonnes
    axes[0].imshow(image) # voir la photo
    axes[0].set_title('Image') #Un titre dans la première colonne
    axes[1].set_title('Histogram') #Un titre dans la deuxième colonne
    axes[1].plot(exposure.histogram(image[...,0])[0], color='red') #La courbe de l'image est roug
e dans la deuxième colonne
    axes[1].plot(exposure.histogram(image[...,1])[0], color='green') #La courbe de l'image est ve
rt dans la deuxième colonne
    axes[1].plot(exposure.histogram(image[...,2])[0], color='blue') #La courbe de l'image est bleu
 dans la deuxième colonne
    axes[1].set_xlim([1, 256]) #longueur d'axe x
    axes[1].set_ylim([0, 5000])#longueur d'axe y
histColor(image) # pour afficher les résulats
```

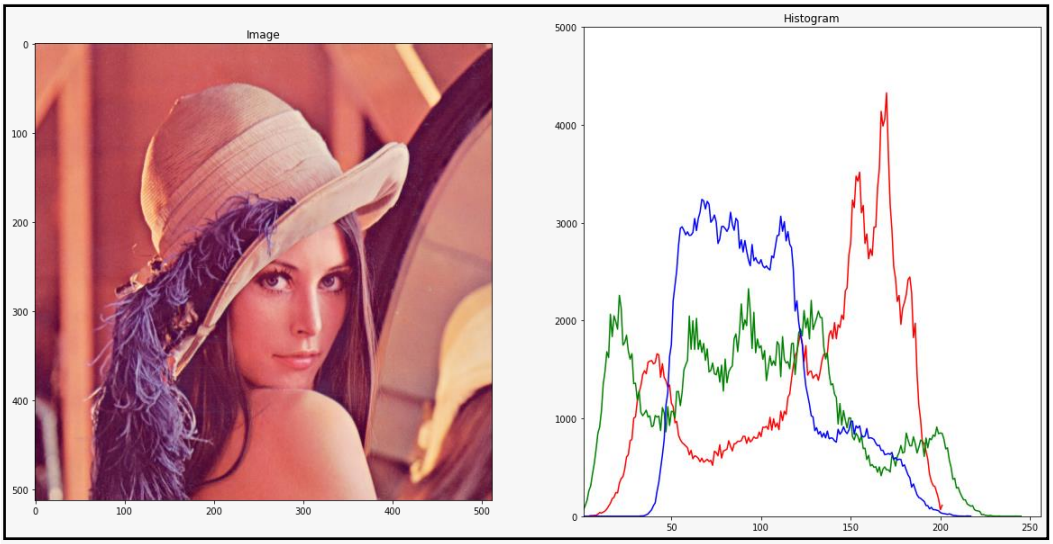


Figure III.13 : L'image de Lena avec son histogramme RVB

Aussi le code d'histogramme de niveau de gris :

```
plt.hist(image.ravel(),bins = 256, range = [0,256])
plt.show()
```

plt.hist(x, bins=, range=)

paramètres :

- **bins :** Ce paramètre est un paramètre facultatif et il contient l'entier ou la séquence ou la chaîne.
- **range :** Ce paramètre est un paramètre facultatif et c'est la plage inférieure et supérieure des bins.

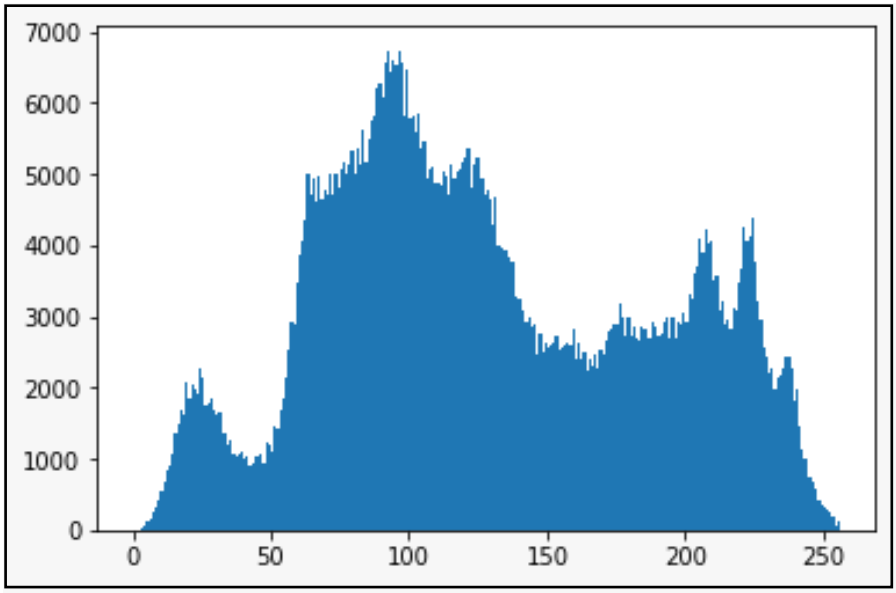


Figure III.14 : L'histogramme de niveau de gris

III.7. Le bruitage de l'image

Nous utiliserons deux types de bruit pour une meilleure appréciation des résultats des filtres appliqués.

III.7.1. Bruit gaussien

La fonction de bruit gaussien est :

```
def GaussNoise(image, var=1000): #Définition de la fonction #variance
    mean = 0 #moyenne
    sigma = var ** 0.5
    gauss = np.random.normal(mean, sigma, image.shape)

    res = image + gauss
    noisy = np.clip(res, 0, 255).astype(np.uint8)
    return noisy
```

np.random.normal(loc = 0.0, scale = 1.0, size = None) : crée un tableau de forme spécifiée et le remplit avec des valeurs aléatoires qui font en fait partie de Normal (Gaussian) Distribution. Cette distribution est également connue sous le nom de courbe en cloche en raison de sa forme caractéristique.

Paramètres :

- **loc :** [float ou array_like] Moyenne de la distribution.
- **scale :** [float ou array_like] Dérivation standard de la distribution.
- **size :** [tuples int ou int].

np.clip(a, a_min, a_max) : est utilisée pour couper (limiter) les valeurs dans un tableau.

Paramètres :

- **a :** Tableau contenant les éléments à découper.
- **a_min :** valeur minimale.
- **a_max :** valeur maximale.

```
imageGN = GaussNoise(image, var=1000)

plt.figure(figsize=(20,10)),plt.subplot(121),plt.imshow(image),plt.title('Image Originale')pl
t.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(imageGN),plt.title('Bruit de Gauss')
plt.xticks([]), plt.yticks([])
plt.show()
```

Pour afficher l'image bruitée :

Figure III.15 : L'image de Lena avec bruit de Gauss

Figure III.16 : L'image Université avec bruit de Gauss

Nous allons changer la variance du bruit de gauss (les valeurs sont 500, 1000, 2000,5000) et afficher les résultats avec les fonctions de la bibliothèque **matplotlib** avec le code suivant :

```
plt.figure(figsize=(20,10)),plt.subplot(141),plt.imshow(imageGN1),plt.title('Bruit de gauss avec var=500')
plt.xticks([]), plt.yticks([])
plt.subplot(142),plt.imshow(imageGN2),plt.title('Bruit de gauss avec var=1000')
plt.xticks([]), plt.yticks([])
plt.subplot(143),plt.imshow(imageGN3),plt.title('Bruit de gauss avec var=3000')
plt.xticks([]), plt.yticks([])
plt.subplot(144),plt.imshow(imageGN4),plt.title('Bruit de gauss avec var=5000')
plt.xticks([]), plt.yticks([])
plt.show()
```

- **plt.figure :** les dimensions de l'image de sortie.
- **plt.supplot :** le nombre de lignes/colonnes de la grille des sous-parcelles.
- **plt.imshow :** l'image de sortie.
- **plt.title :** Titre du la grille des sous-parcelles.
- plt.xticks,plt.yticks : numérotation des axes.

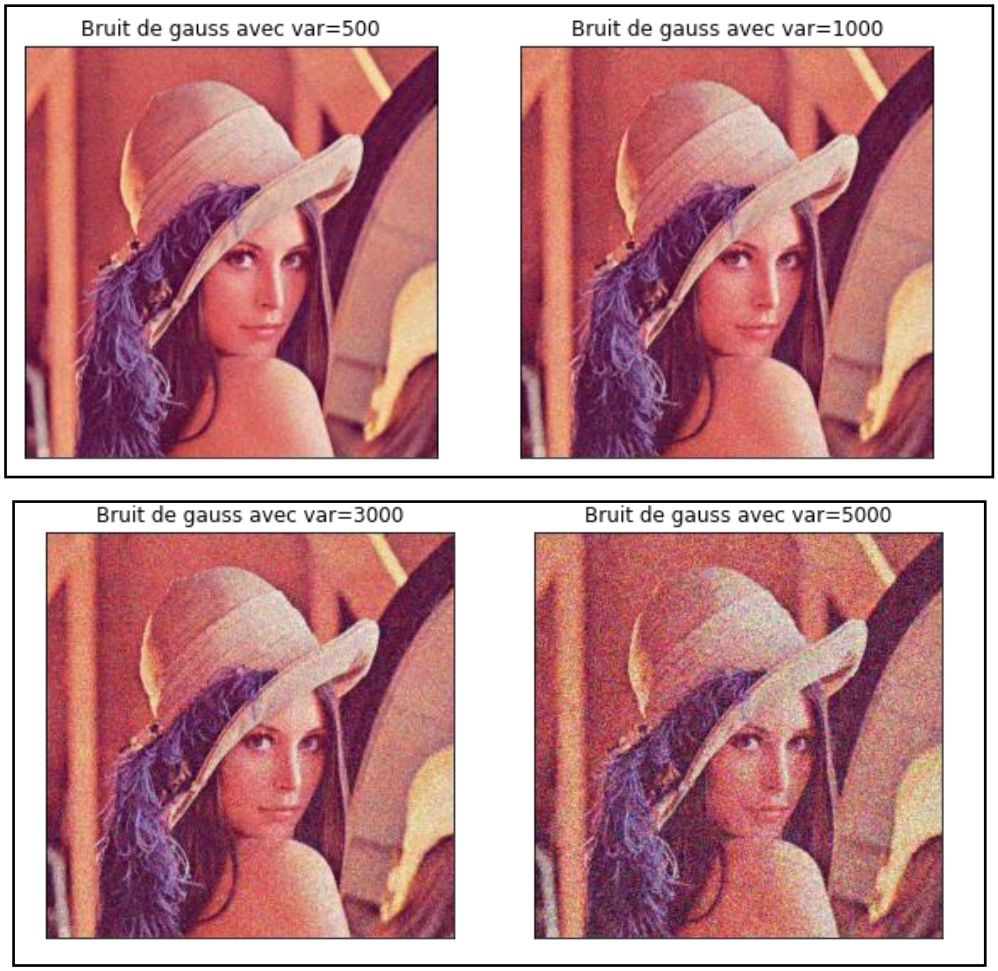


Figure III.17 : Bruit de gauss avec des variances différentes

III.7.2. Bruit sel et poivre

La fonction de bruit sel et poivre est :

```
def Sel_et_Poivre(image, densité=0.04):#Définition de la fonction #densité de bruit
    out = np.copy(image)
    #Mode sel  Pour afficher les points de bruit blanc
    mask = np.random.rand(image.shape[0], image.shape[1]) <= densité / 2
    out[mask] = 255
    # Mode Poivre Pour afficher les points de bruit noir
    mask = np.random.rand(image.shape[0], image.shape[1]) <= densité / 2
    out[mask] = 0
    return out
```

np.copy() : nous pouvons faire une copie de tous les éléments de données présents dans la matrice. Si nous modifions un élément de données dans la copie, cela n'affectera pas la matrice d'origine.

np.random.rand() : crée un tableau de forme spécifiée et le remplit avec des valeurs aléatoires.

Pour afficher l'image bruitée :

```
imageSP = Sel_et_Poivre(image, densité=0.15)
plt.figure(figsize=(20,10)),plt.subplot(121),plt.imshow(image),plt.title('Image Originale')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(imageSP),plt.title('Sel et Poivre')
plt.xticks([]), plt.yticks([])
plt.show()
```

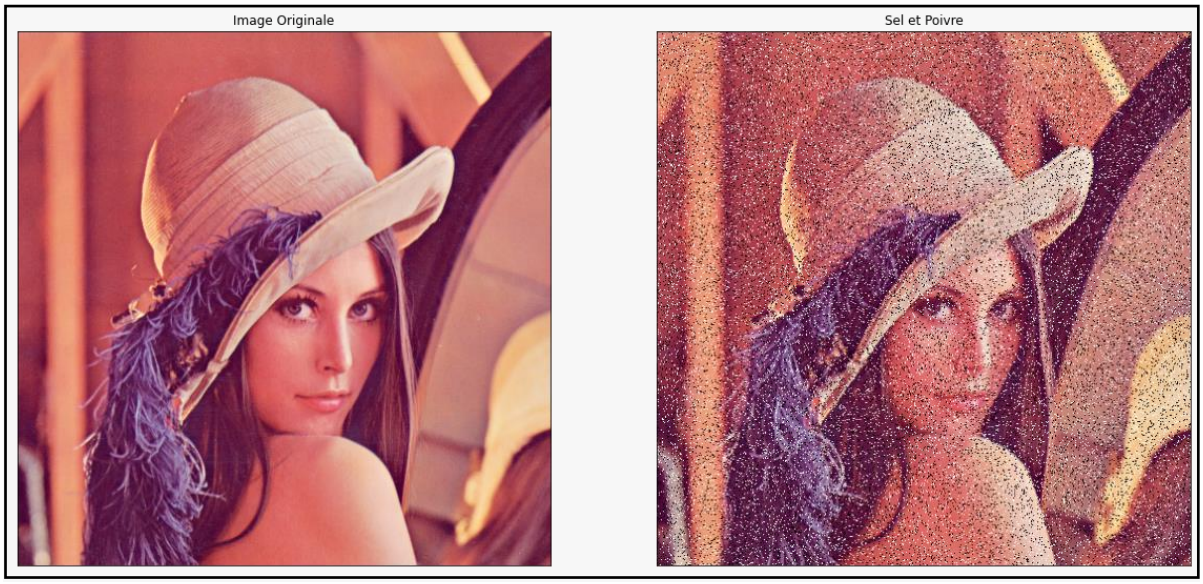


Figure III.18 : L'image Lena avec bruit de Sel et Poivre

Figure III.19 : L'image Université avec bruit de Sel et Poivre

Nous allons changer la densité du bruit de sel et poivre (les valeurs sont 0.05, 0.10, 0.20, 0.50) et afficher les résultats avec les fonctions de la bibliothèque **matplotlib.**

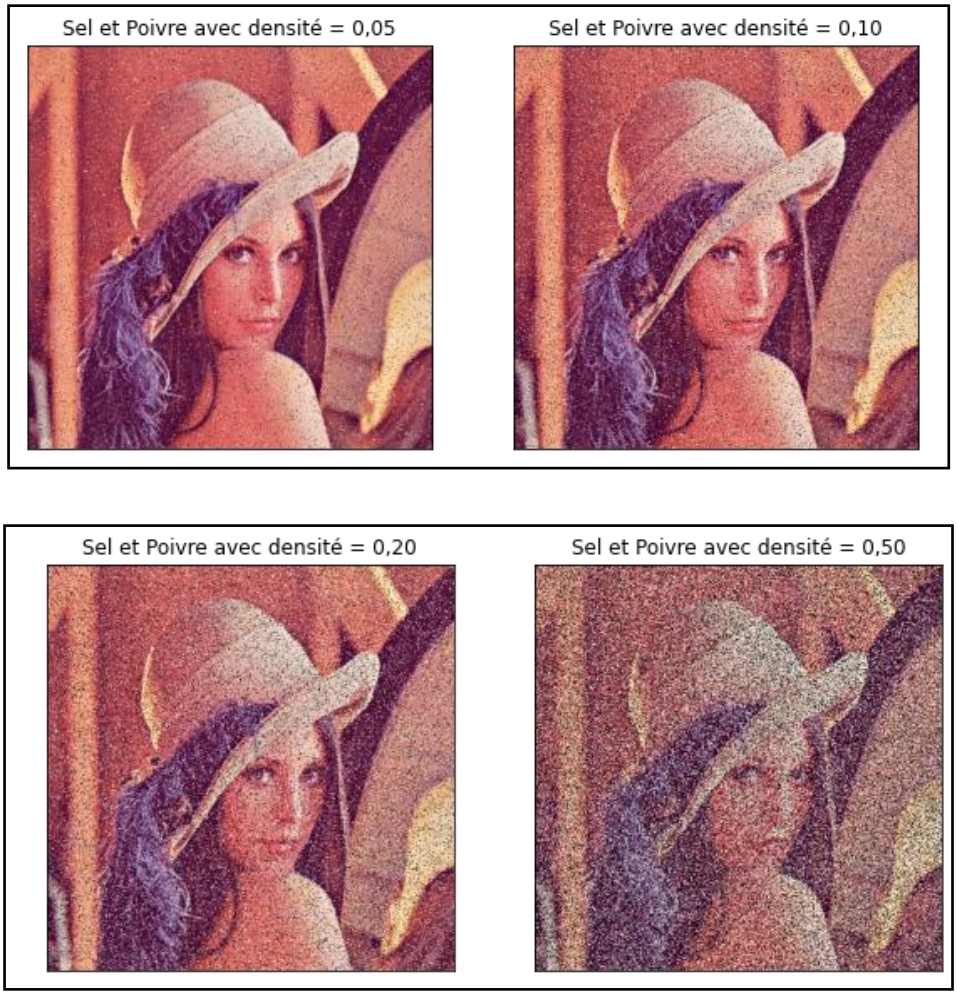


Figure III.20 : Bruit de sel et poivre avec des densités différentes

III.8. Filtrage d'image

III.8.1. Filtre gaussien

La syntaxe de ce filtre est :

dst = cv2.GaussianBlur(src ,size(int, int) , sigma)

Paramètres :

- **dst :** image de sortie.
- **src :** image d'entrée.
- **size(int, int) :** la taille du noyau (la hauteur et la largeur).
- **sigma :** L'écart type du noyau gaussien.

Nous filtrons l'image de bruit des deux types en entrant le nom de l'image de l'entrée (**imageGN** pour bruit de gauss et **imageSP** pour sel et poivre bruit).

```
Gaussfiltre = cv2.GaussianBlur(imageGN,(3,3),-1)
Gaussfiltre = cv2.GaussianBlur(imageSP,(3,3),-1)
```

Remarque 1 : Chaque fois que nous changeons la taille du noyau, nous gardons le sigma constant.

Remarque 2 : Les valeurs utilisées dans les deux bruits sont : Var = 1000 pour bruit de gauss, Densité=0.15 pour bruit de sel et poivre.

Toujours afficher le résultat en utilisant les fonctions de la bibliothèque **matplotlib.**

La fonction pour afficher le PSNR d'image est :

```
def psnr(target, ref): #Target=image cible   #ref=image de référence
    # Assume target is RGB/BGR image
    target_data = target.astype(np.float32)
    ref_data = ref.astype(np.float32)
    diff = ref_data - target_data
    diff = diff.flatten('C')
    rmse = np.sqrt(np.mean(diff ** 2.))
    return 20 * np.log10(255. / rmse)
psnr(imageSP,image)
```

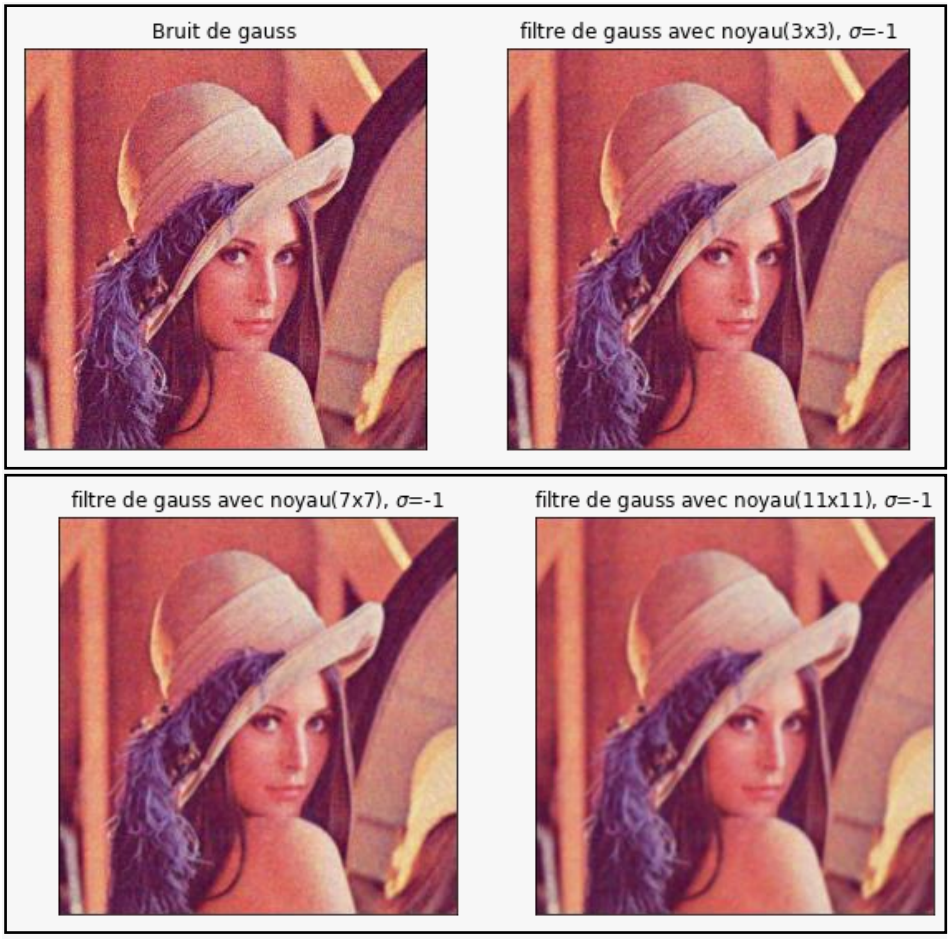


Figure III.21 : Résultats du filtre de gauss pour l'image Lena bruitée avec gauss

Figure III.22 : Résultats du filtre de gauss pour l'image Lena bruitée avec sel et poivre

PSNR	Bruit	Filtre de gauss avec noyau (3x3)	Filtre de gauss avec noyau (7x7)	Filtre de gauss avec noyau (11x11)
Bruit de gauss	18.432844983056405	20.221310723556382	18.688395690312646	18.267047937555315
Bruit sel et poivre	13.570773821422028	15.572316082029582	14.148936049211274	13.680861782982781

Tableau III.1 : Résultats du PSNR du filtre de gauss pour l'image lena

- **L'analyse de la performance de filtre gaussien**

Pour l'image bruitée en gauss, on note que le meilleur résultat visuel avec le masque 7x7 et 11x11 mais visuellement il y a un effet de flou dans le masque 11x11. Quant au bruit du sel et poivre, il n'a pas pu supprimer le bruit totalement, seulement il y a un effet de flou.

III.8.2. Filtre median

La syntaxe de ce filtre est :

dst = cv2.medianBlur(src , kernel size)

Paramètres :

- **dst :** image de sortie.
- **src :** image d'entrée.
- **kernel size :** la taille du noyau car il prend toujours une matrice carrée, la valeur doit être un entier positif supérieur à 2.

```
median = cv2.medianBlur(imageGN,3)
median = cv2.medianBlur(imageSP,3)
```

Toujours afficher le résultat en utilisant les fonctions de la bibliothèque **matplotlib.**

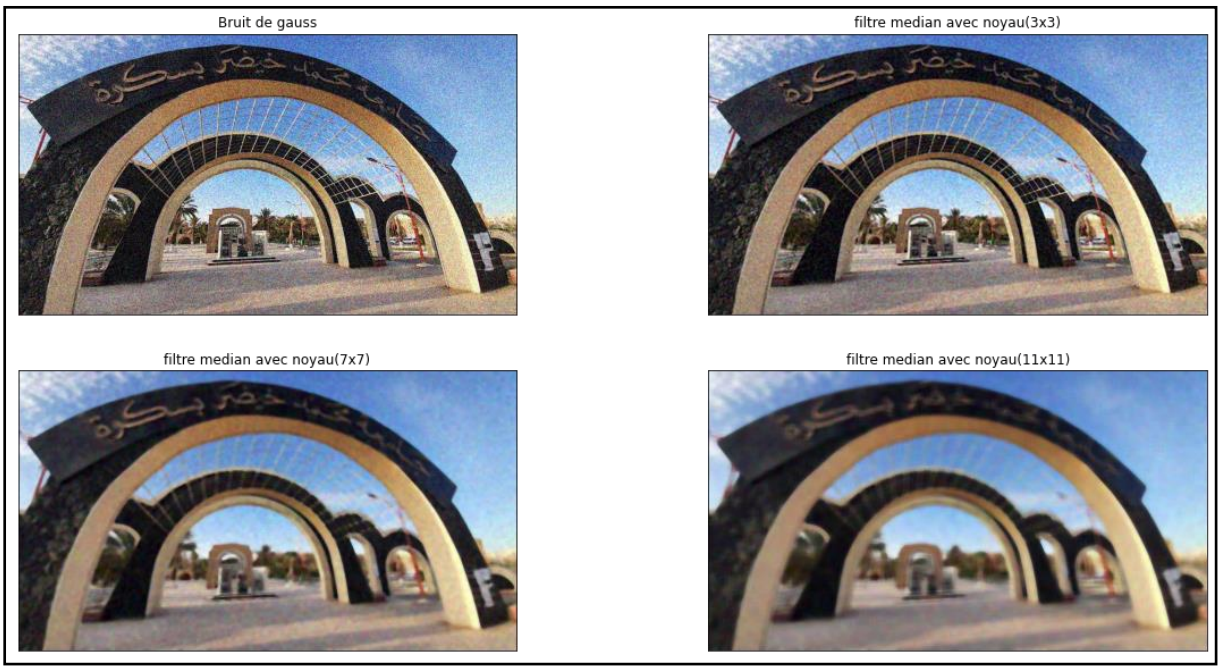


Figure III.23 : Résultats du filtre médian pour l'image Université bruitée avec gauss

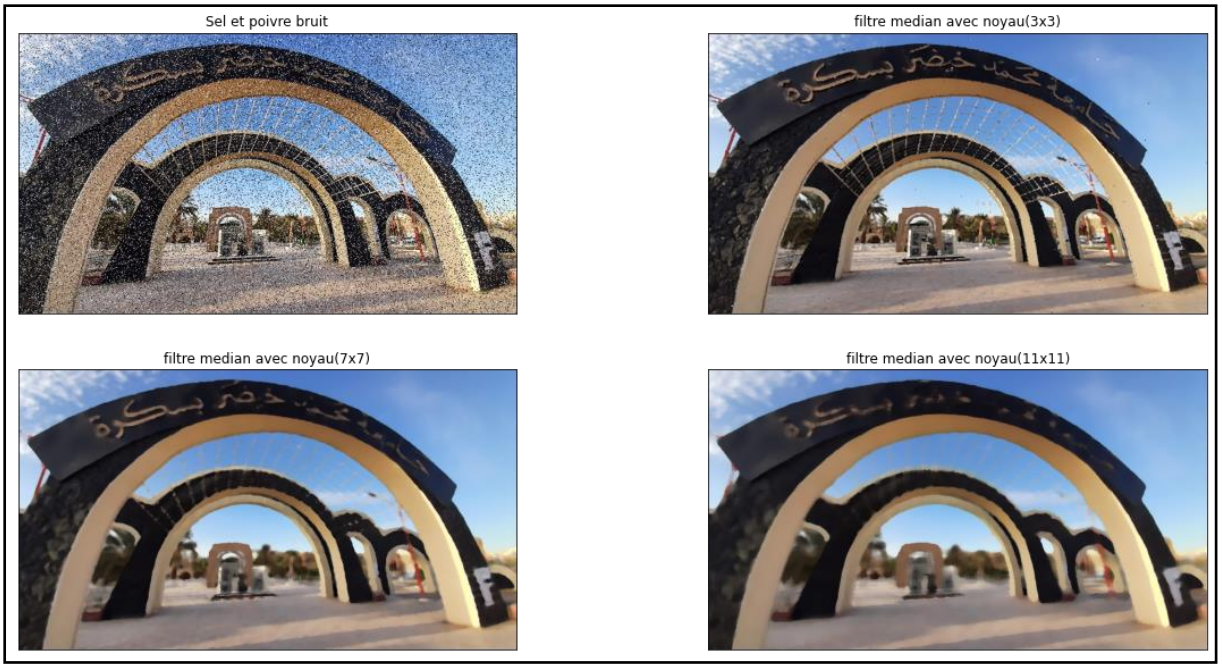


Figure III.24 : Résultats du filtre médian pour l'image de Université bruitée avec sel et poivre

PSNR	Bruit	Filtre median avec noyau (3x3)	Filtre median avec noyau (7x7)	Filtre median avec noyau (11x11)
Bruit de gauss	18.432844983056405	18.546573005222932	18.054360112016532	17.80117383560192
Bruit sel et poivre	13.570773821422028	13.625170283944254	13.492581301826544	13.427360359820783

Tableau III. 2: Résultats du PSNR du filtre median pour l'image de Université

- **L'analyse de la performance de filtre**

Dans les deux images bruitées, le filtre Médian nous a donné les meilleurs résultats et un effet de flou dans le masque 11x11 (meilleur résultat pour bruit de sel et poivre dans le masque 3x3

et pour bruit de gauss le masque 7x7), parce que le filtre Médian améliore les images dégradées, en supprimant les pixels présentant des valeurs aberrantes dues au bruit.

III.8.3. Filtre moyenneur

La syntaxe de ce filtre est :

dst = cv2.blur(src , kernel size)

Paramètres :

- **dst :** image de sortie.
- **src :** image d'entrée.
- **kernel size :** la taille du noyau

```
moyen = cv2.blur(imageGN,(3,3))#imageGN pour le bruit de gauss
moyen = cv2.blur(imageSP,(3,3))#imageSP pour le sel et poivre bruit
```

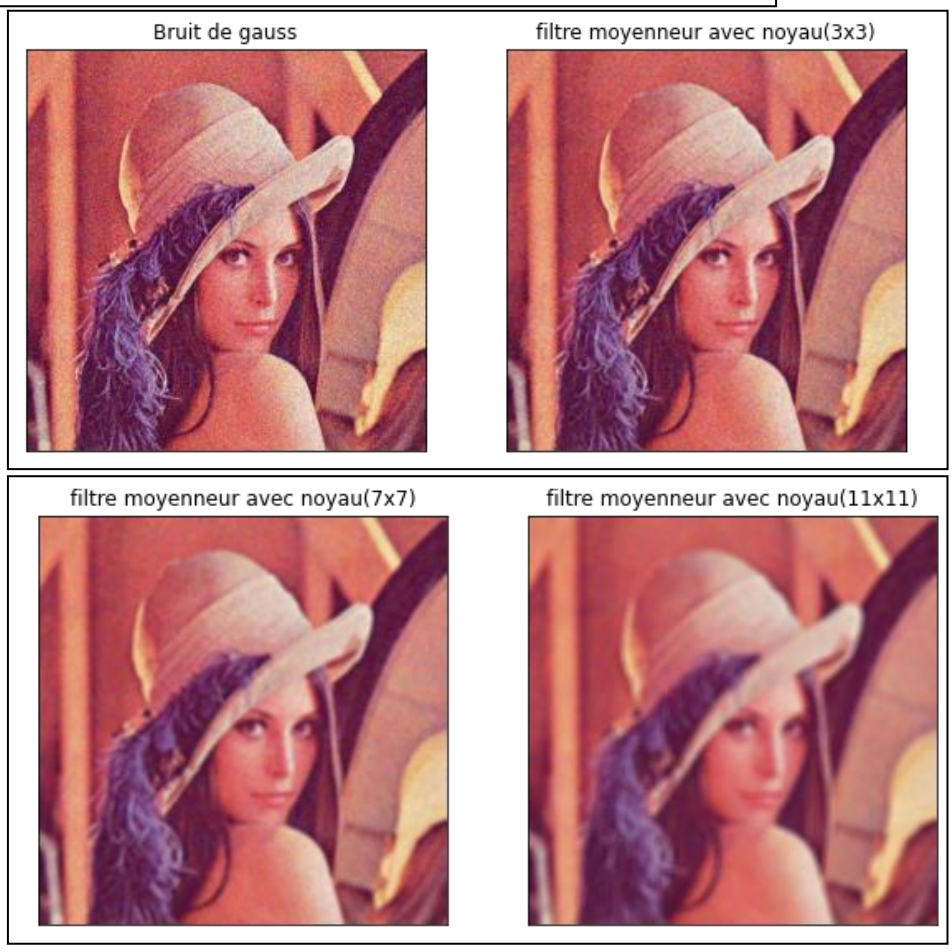


Figure III. 25 : Résultats du filtre moyenneur pour l'image de lena bruitée avec gauss

Figure III.26 : Résultats du filtre moyenneur pour l'image Lena bruitée avec sel et poivre

PSNR	Bruit	Filtre median avec noyau (3x3)	Filtre median avec noyau (7x7)	Filtre median avec noyau (11x11)
Bruit de gauss	18.432844983056405	18.78325371522877	18.031995473971367	17.699672437551634
Bruit sel et poivre	13.570773821422028	14.144174695435938	13.63718153691283	13.504043258379061

Tableau III. 3 : Résultats du PSNR du filtre moyenneur pour l'image de lena

- **L'analyse de la performance de filtre**

Visuellement, on voit que ce filtre ne fait que du lissage et ne traite pas les pixels de bruit comme un filtre Médian (le meilleur résultat visuel avec le masque 7x7 avec un peu flou).

III.8.4. Filtre bilatéral

La syntaxe de ce filtre est :

dst = cv2.bilateralFilter(src , d , sigmaColor , sigmaSpace)

Paramètres :

- **dst :** image de sortie.
- **src :** image d'entrée.
- **d :** Diamètre de chaque pixel.
- **sigmaColor :** valeur de sigma dans l'espace colorimétrique. Plus la valeur est élevée, plus les couleurs éloignées les unes des autres commenceront à se mélanger.
- **sigmaSpace :** valeur de sigma dans l'espace des coordonnées. Plus sa valeur est élevée, plus les pixels se mélangeront, étant donné que leurs couleurs se situent dans la plage sigmaColor.

Nous allons appliquer le filtre avec une valeur diamètre constante et modifier la valeur de sigma (d=30 , sigmaColor\sigmaSpace=(50,100,200))

```
bilafiltre= cv2.bilateralFilter(imageGN,30,50,50)
bilafiltre= cv2.bilateralFilter(imageSP,30,50,50)
```

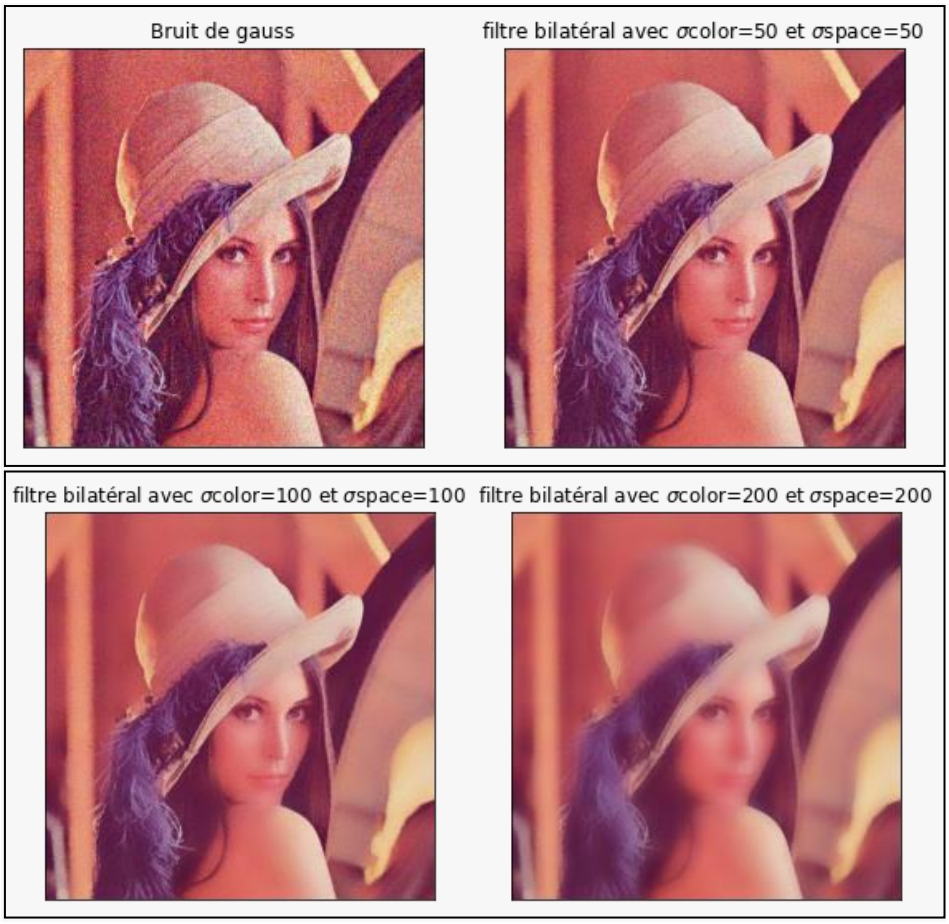


Figure III.27 : Résultats du filtre bilatéral pour l'image Lena bruitée avec gauss

Figure III.28 : Résultats du filtre bilatéral pour l'image de lena bruitée avec sel et poivre

PSNR	Bruit	Filtre median avec σ=50	Filtre median avec σ=100	Filtre median avec σ=200
Bruit de gauss	18.432844983056405	24.103448489866537	20.058823899642192	18.154581305842765
Bruit sel et poivre	13.570773821422028	32.522085666130366	27.076670941907008	18.555972101712648

Tableau III. 4: Résultats du PSNR du filtre bilatéral pour l'image de lena

- **L'analyse de la performance de filtre**

Nous pouvons voir que plus la valeur sigma est élevée, plus l'image sera lissée, tout en préservant les bords et en réduisant le bruit (pas entièrement pour le bruit de sel et poivre).

III.8.5. Filtres morphologique

a. Dilatation et Erosion

La syntaxe de dilatation est :

dst = cv2.dilate(src , kernel, iterations)

Paramètres :

- **dst :** image de sortie.
- **src :** image d'entrée.
- **kernel :** élément structurant utilisé pour la dilatation. Un élément structurant rectangulaire 3x3 est utilisé.
- **iterations :** C'est le nombre de fois où la dilatation est appliquée.

La syntaxe d'érosion est :

dst = cv2.erode(src , kernel, iterations)

Paramètres :

- **kernel :** élément structurant utilisé pour l'érosion. Un élément structurant rectangulaire 3x3 est utilisé.
- **iterations :** C'est le nombre de fois où l'érosion est appliquée.

Dans ce filtre nous allons l'utiliser sur 3 images (l'image d'origine, l'image de bruit de gauss et l'image de sel et poivre bruit).

```
kernel = np.ones((3,3) , np.uint8) #la taille de noyau
#dilatation
dilateIMG = cv2.dilate(imageGN ,kernel , iterations = 1)
#Erosion
erodeIMG = cv2.erode(imageGN ,kernel , iterations=1)
```

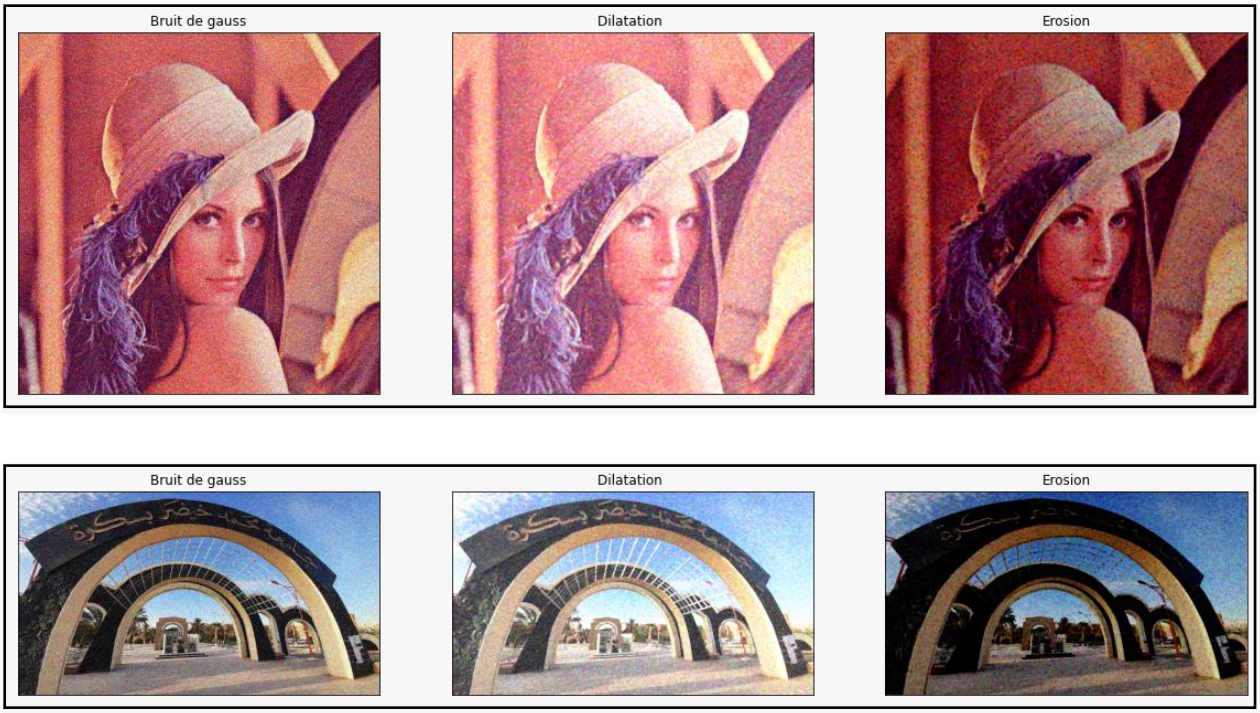


Figure III.29 : Résultats du la dilatation et l'érosion pour les deux images bruitées avec gauss

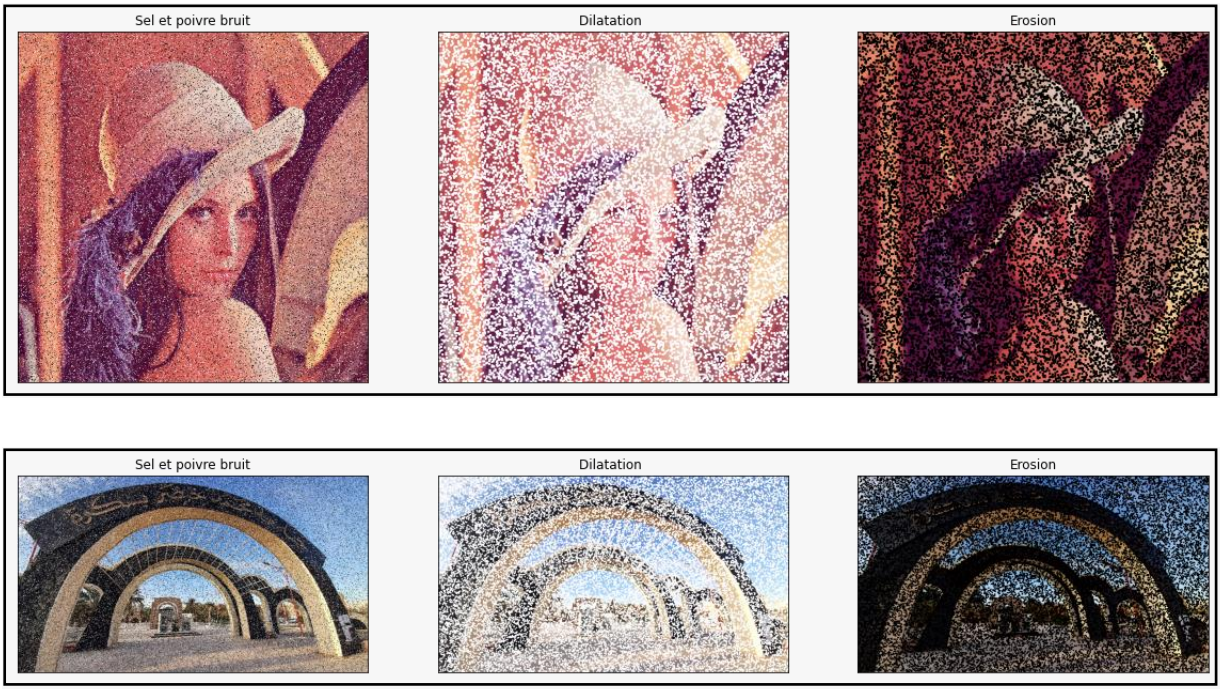


Figure III.30 : Résultats du la dilatation et l'érosion pour les 2 images bruitées avec sel et poivre

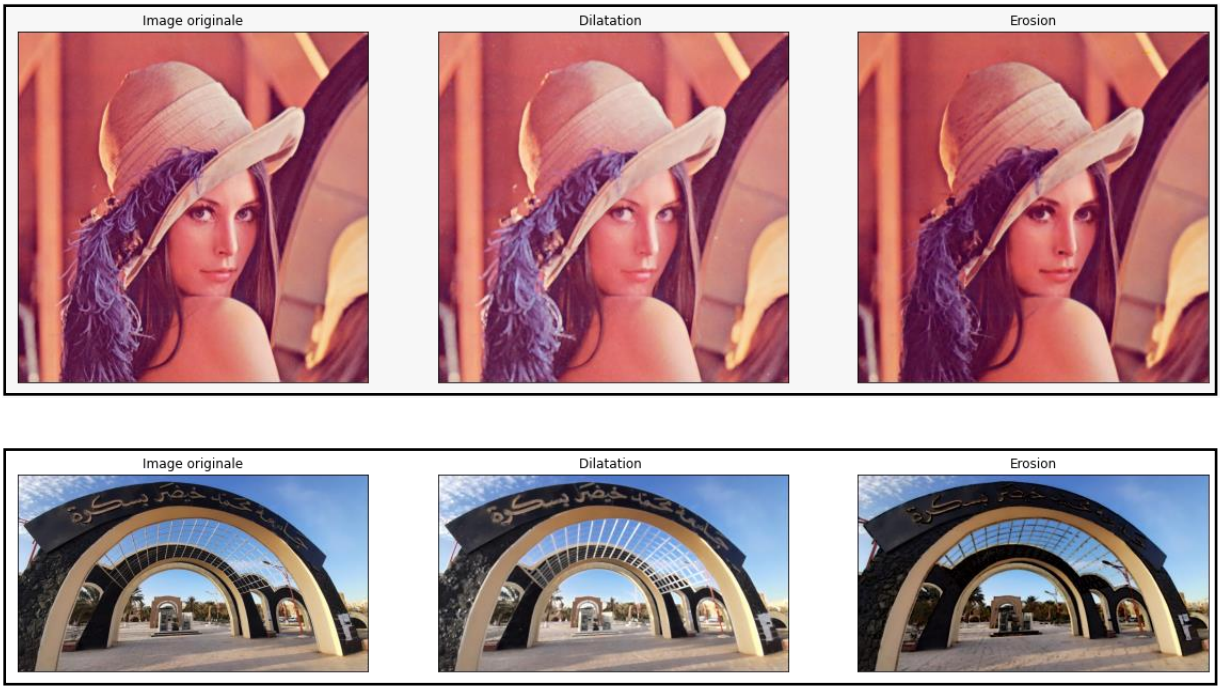


Figure III.31 : Résultats du la dilatation et l'érosion pour les deux images

b. Ouverture et Fermeture

La syntaxe d'ouverture est :

dst = cv2.morphologyEx(src, cv2.MORPH_OPEN, kernel)

Parameters:

- **dst :** image de sortie.
- **src :** image d'entrée.
- **cv2.MORPH_OPEN**: application de l'opération d'ouverture Morphologique.
- **kernel**: élément structurant.

La syntaxe de fermeture est :

dst = cv2.morphologyEx(src, cv2.MORPH_CLOSE, kernel)

Parameters:

- **cv2.MORPH_CLOSE:**application de l'opération de fermetureMorphologique.

Dans ce filtre aussi nous allons l'utiliser sur 3 images (l'image d'origine, l'image de bruit de gauss et l'image de sel et poivre bruit).

```
kernel = np.ones((3,3) , np.uint8)
#ouverture
ouvertureIMG = cv2.morphologyEx(imageSP, cv2.MORPH_OPEN, kernel)
#fermeture
fermetureIMG = cv2.morphologyEx(imageSP, cv2.MORPH_CLOSE, kernel)
```

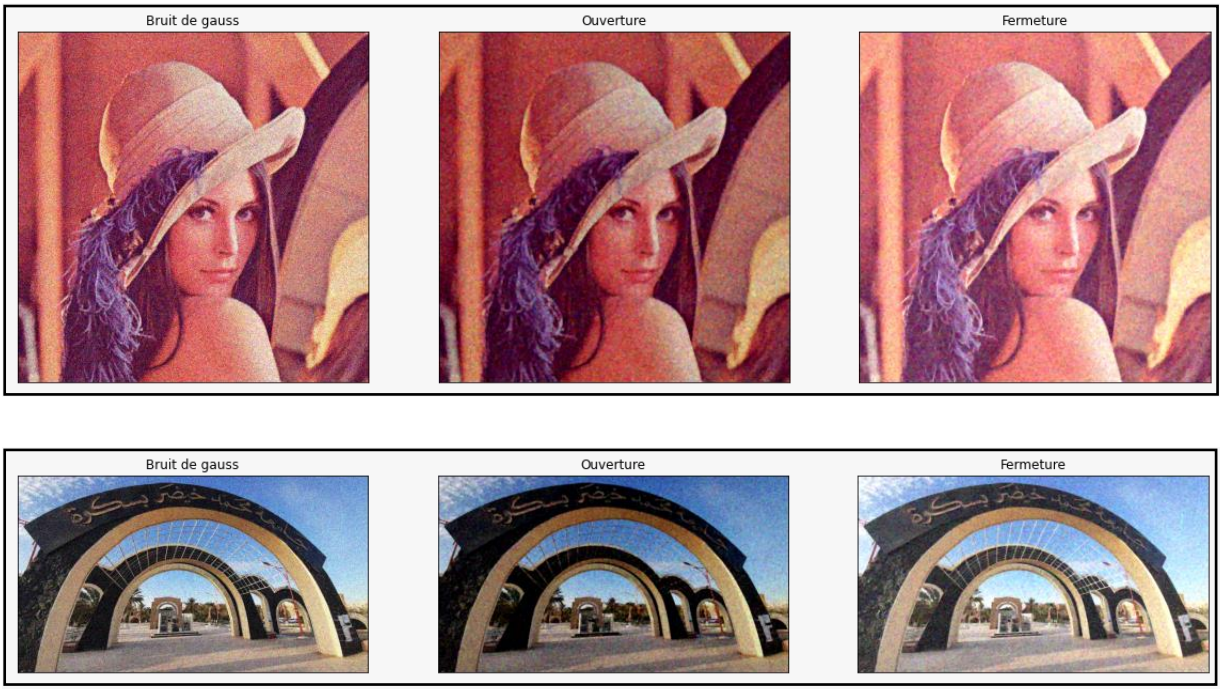


Figure III.32 : Résultats du l'ouverture et la fermeture pour les deux images bruitées avec gauss

Figure III.33 : Résultats du l'ouverture et la fermeture pour les 2 images bruitées avec sel et poivre

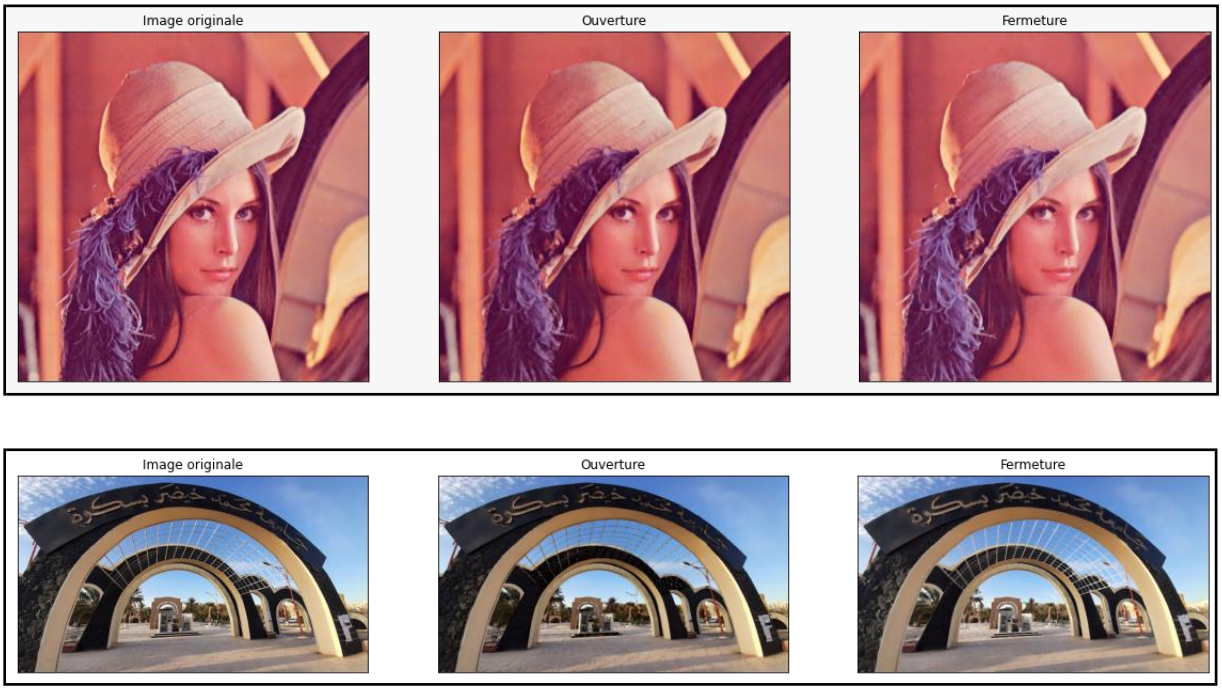


Figure III.34 : Résultats du l'ouverture et la fermeture pour les 2 images

- **L'analyse de la performance de filtre**

Pour la dilatation, on note qu'elle ne filtre pas le bruit, mais elle augmente la luminosité de l'image et augmente la taille des objets, et pour l'érosion, A l'opposé la dilatation, il réduit la luminosité de l'image et réduit la taille des objets.

III.8.6. Filtre de canny

Ce filtre permet de définir les contours de l'image. Nous l'utiliserons uniquement sur l'image d'origine.

La syntaxe de ce filtre est :

```
dst = cv2.Canny(src, T_lower, T_upper)
```

Parameters:

- **T_lower :** valeur de seuil inférieure.
- **T_upper :** valeur de seuil supérieure (trois fois le seuil inférieur suivant la recommandation de Canny).

```
CannyImg = cv2.Canny(image,100,100)
```

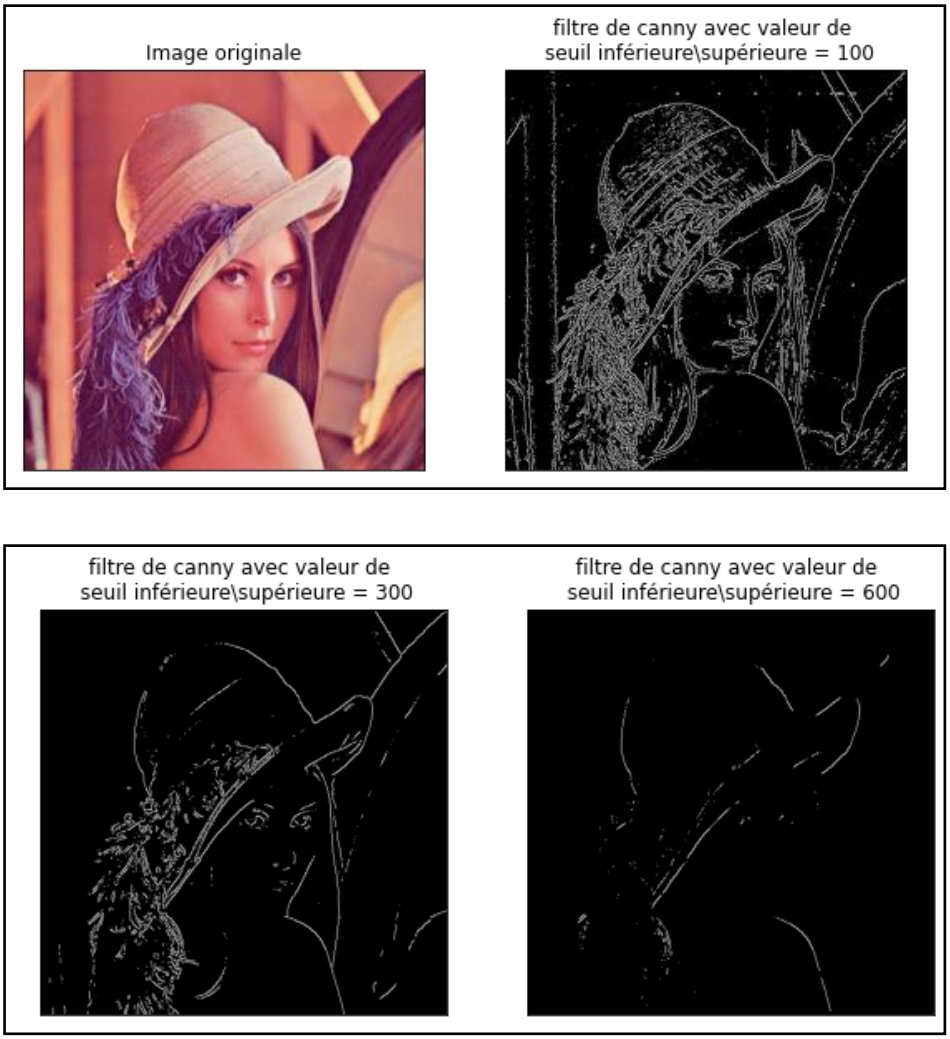


Figure III.35 : Résultats du filtre de canny pour l'image Lena

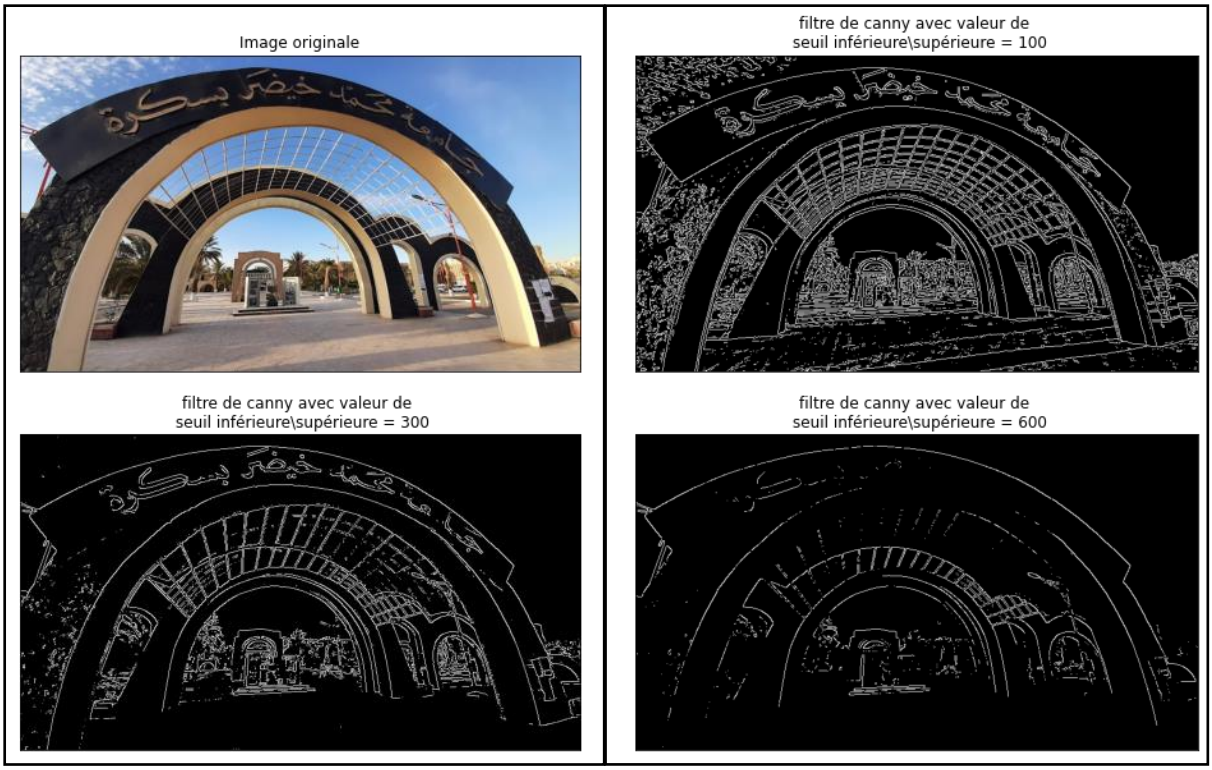


Figure III.36 : Résultats du filtre de Canny pour l'image de Université

- **L'analyse de la performance de filtre**

Pour le filtre Canny, Tout ce que nous réduisons la valeur de seuil inférieure\supérieure nous avons trouvé un meilleur résultat

III.8.7. Filtre Laplacien

La syntaxe de ce filtre est :

dst = cv2.Laplacian(src, ddepth, ksize)

Parameters:

- **ddepth :** La profondeur de l'image résultante traitée, nous remplissons généralement -1, qui est la même profondeur que l'image d'origine. Mais ici, nous devons remplir cv2.CV_64F. En termes simples, si nous remplissons -1, le nombre négatif que nous générons lors du calcul du gradient sera par défaut à 0 par le programme, ce qui se traduira par un côté du bord qui ne peut pas être quitté. cv2.CV_64F a une plage plus large et peut contenir des nombres négatifs.

```
laplacian = cv2.Laplacian(image3,cv2.CV_64F,ksize=3)
laplacian =np.uint8(np.absolute(laplacian))
#On renvoit l'imageàuint8pour nous concentrer sur le centre de matrice.
```

Convertir l'image d'origine en gris.

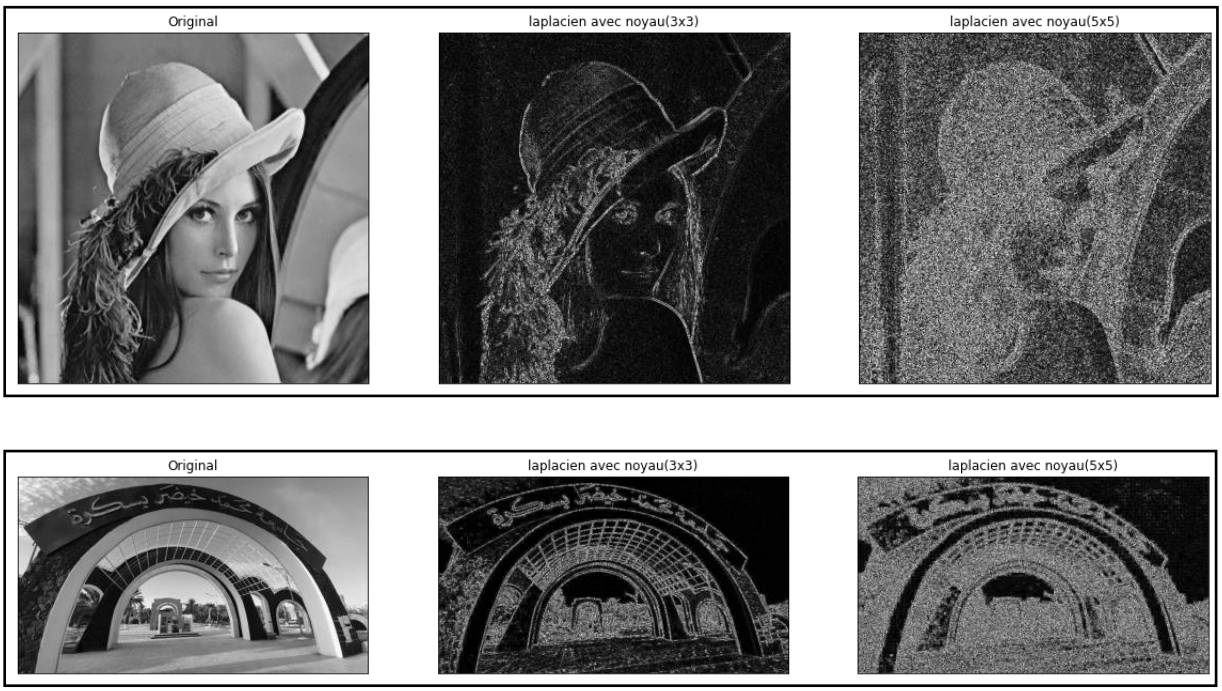


Figure III.37 : Résultats du filtre Laplacien pour les 2 images

Une autre syntaxe pour crée le filtre est :

dst = cv2.filtre2D(src, ddepth, ksize)

Nous allons écrire les trois noyaux du filtre et les appliquer à une image en niveaux de gris.

```
Kernel4 = np.array([[0,1,0],[1,-4,1],[0,1,0]])#Laplacien discret -4
Kernel8 = np.array([[1,1,1],[1,-8,1],[1,1,1]])#Laplacien discret -8
kernelR = np.array([[1,-2,1],[-2,4,-2],[1,-2,1]])#Laplacien de robinson
```

```
dst4 = cv2.filter2D(image,-1,kernel4)
dst8= cv2.filter2D(image,-1,kernel8)
dstR = cv2.filter2D(image,-1,kernelR)
```

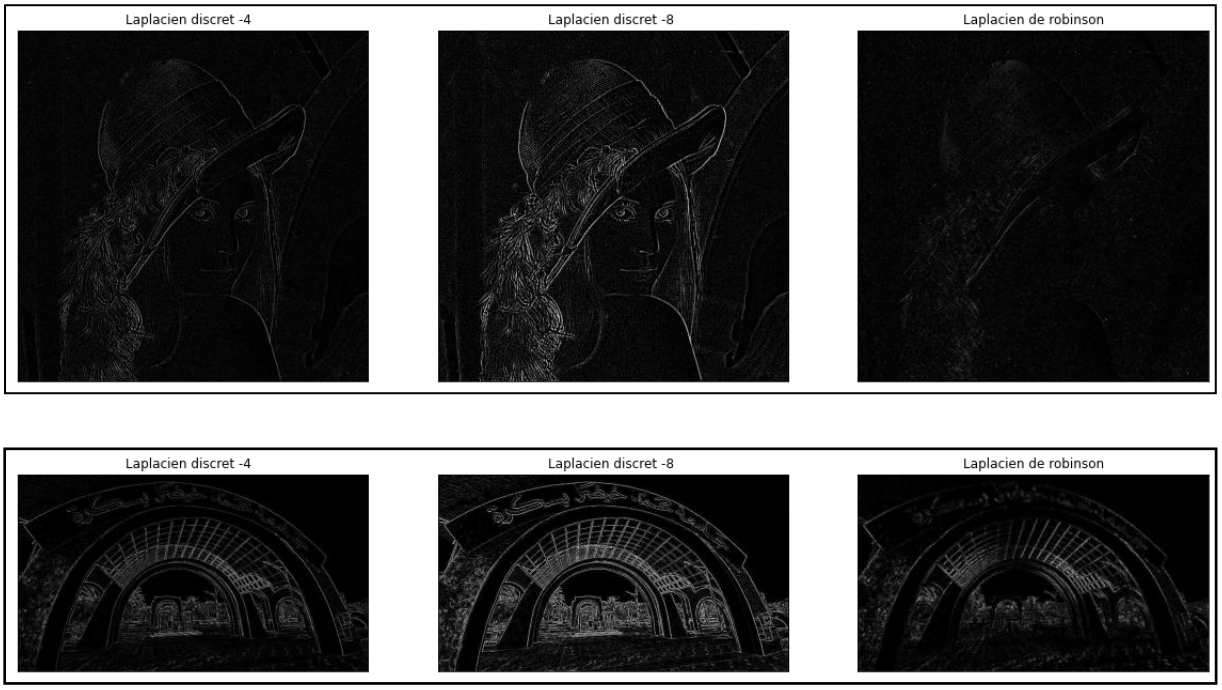


Figure III.38 : Résultats du filtre Laplacien avec trois noyaux pour les deux images

- **L'analyse de la performance de filtre**

Le meilleur résultat de filtre Laplacien pour le masque 3x3 avec le noyau Laplacien discret 8.

III.8.8. Filtres de SOBEL, PREWITT, ROBERTS et KIRSCH:

a. Filtre de SOBEL

La syntaxe de ce filtre est :

dst = cv2.Sobel(src, ddepth, dx, dy, ksize)

Parameters:

dx : Degré de la dérivée x

dy : Degré de la dérivée y

Remarque 1:

(dx, dy) = (1, 0) Détection de contour horizontal.

(dx, dy) = (0, 1) Détection de contour vertical.

(dx, dy) = (1, 1) Détection de contour dans la direction diagonale supérieure droite.

```
sobelx = cv2.Sobel(image,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(image,cv2.CV_64F,0,1,ksize=3)
imgSX=cv2.convertScaleAbs(sobelx)
imgSY=cv2.convertScaleAbs(sobely)
```

Remarque 2:

Nous devons également faire attention à une chose. Lorsque nous calculons les poids dans les directions x et c respectivement, ils contiennent des nombres négatifs. Nous devons prendre la valeur absolue de ces nombres négatifs, sinon il sera toujours classé comme 0. La fonction pour prendre la valeur absolue est **cv2.convertScaleAbs()**.

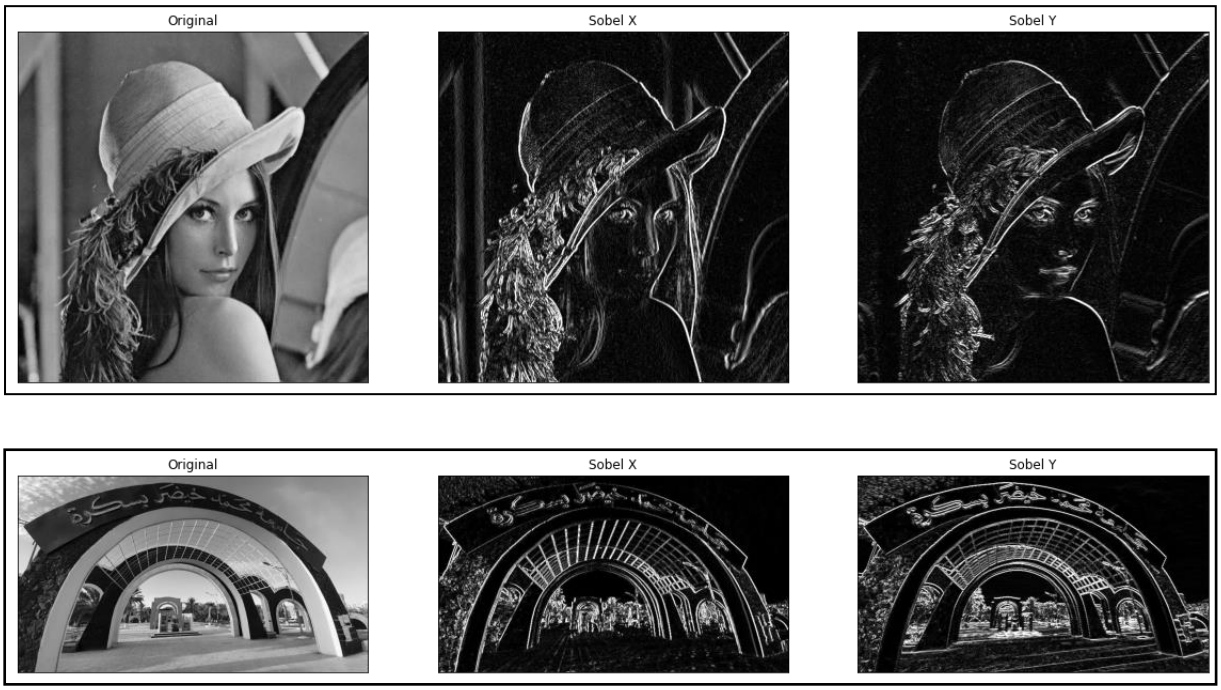


Figure III.39 : Résultats du filtre de sobel pour les 2 images

Remarque 3:

Nous pouvons également créer un filtre de Sobel à travers la fonction **cv2.filtre2D** en écrivant son propre noyau.

```
kernelsobelx = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])#direction horizontale
kernelsobely = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])#direction verticale
```

b. Filtres de PREWITT, ROBERTS et KIRSCH:

Nous les implémenterons à l'aide de la fonction **cv2.filtre2D** en ajoutant leurs propres matrices.

```
#Masques de Prewitt
kernelpx = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])#direction horizontale
kernelpy = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])#direction verticale

#Masques de roberts
kernelrx = np.array([[0,0,0],[0,1,0],[0,0,-1]])#direction horizontale
kernelry = np.array([[0,0,0],[0,0,1],[0,-1,0]])#direction verticale

#Masques de kirsh
kernelkx = np.array([[-3,-3,5],[-3,0,5],[-3,-3,5]])#direction horizontale
kernelky = np.array([[-3,-3,3],[-3,0,-3],[5,5,5]])#direction verticale

prewittx = cv2.filter2D(image,-1,kernelpx)
prewitty = cv2.filter2D(image,-1,kernelpy)

robertx = cv2.filter2D(image,-1,kernelrx)
roberty = cv2.filter2D(image,-1,kernelry)

kirshx = cv2.filter2D(image,-1,kernelkx)
kirshy = cv2.filter2D(image,-1,kernelky)
```

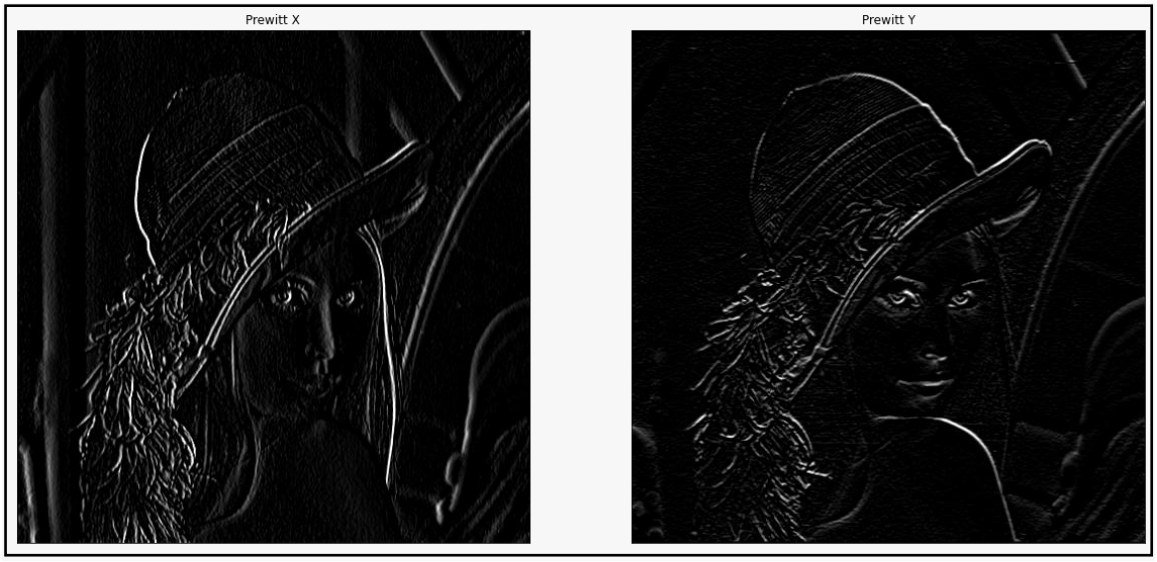


Figure III.40 : Résultats des filtres de PREWITT pour l'image Lena

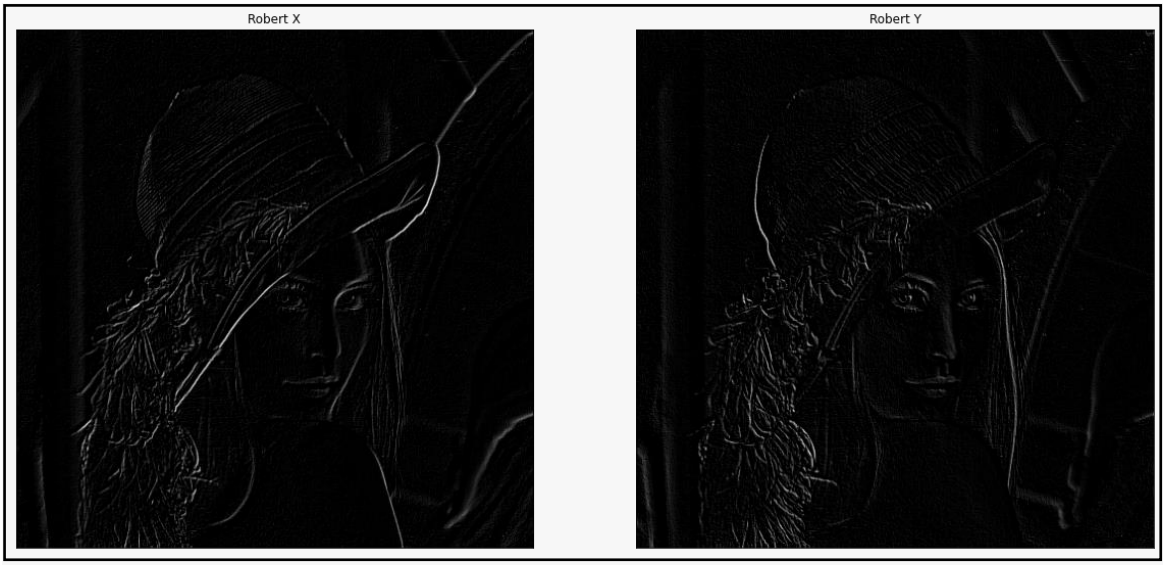


Figure III.41 : Résultats des filtres de ROBERTS pour l'image Lena

Figure III.42 : Résultats des filtres de KIRSCH pour l'image Lena

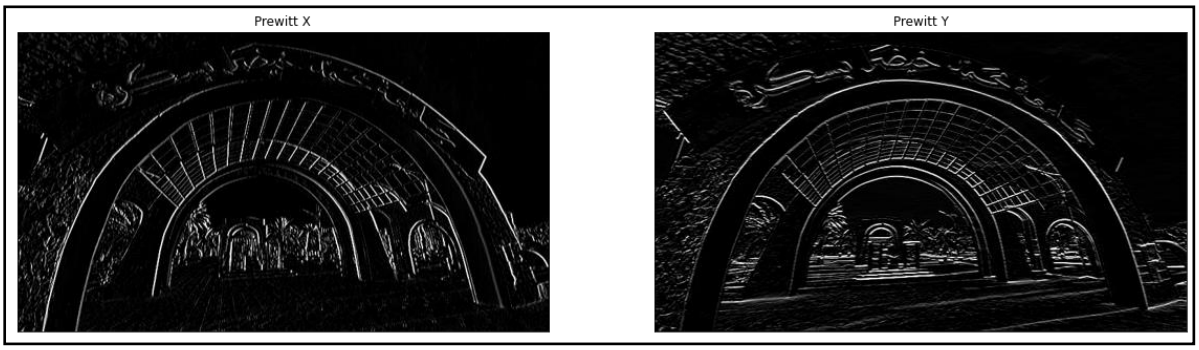


Figure III.43 : Résultats des filtres de PREWITT pour l'image de l'université

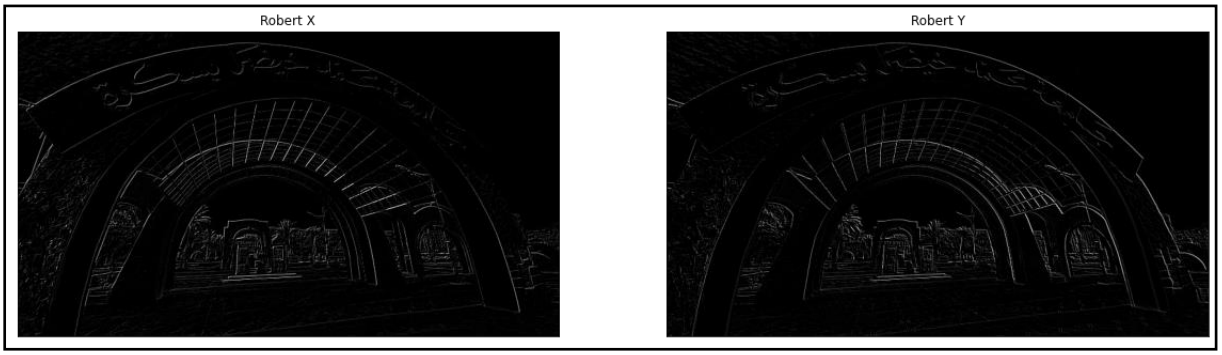


Figure III.44 : Résultats des filtres de ROBERTS pour l'image de l'université

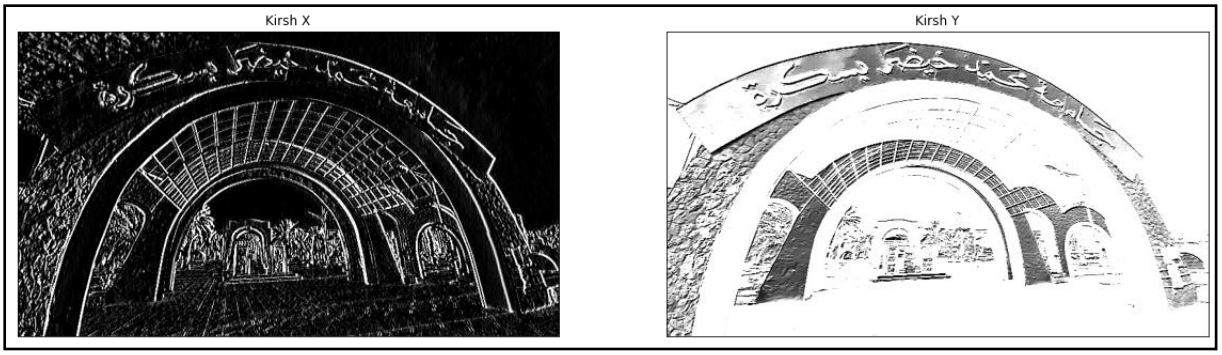


Figure III.45 : Résultats des filtres de KIRSCH pour l'image de l'université

- **L'analyse de la performance de filtre**

Tous ces filtres servent à définir les contours, Le meilleur résultat que nous avons trouvé était les filtres Sobel et Kirsh dans le noyau de détection de contour horizontale.

III.9. Conclusion

Ce chapitre présente plusieurs utilisations de la bibliothèque OpenCV Python par Google Collab pour filtrage d'images, premièrement, nous avons présenté les outils de base pour notre travail et les avons expliqués, deuxièmement, nous avons fait du bruit pour l'image de l'utilisateur. Ensuite, nous avons présenté comment chaque filtre utilisé, et enfin, nous avons parlé du meilleur résultat pour chaque filtre.

Nous avons découvert qu'un filtre était une fonction spéciale qui applique des effets aux images, améliore la qualité de l'image et supprime le bruit. L'application de filtres peut prendre un certain temps, en particulier sur les grandes images.

Conclusion générale

Les images numériques sont affectées par le bruit à cause des chaînes d'acquisition de l'image, ce qui rend la clarté de l'image plus complexe en termes d'analyse d'image et de techniques de détection des contours.

Et à partir de là, en général, les objectifs du filtrage sont de corriger ce défaut pour renforcer l'homogénéité des pixels appartenant à une même région en préservant les contours ainsi que la détection de contours avec de bonne localisation et une rapidité de traitement.

Et à travers ce travail, nous avons vu de nombreux filtres qui permettent de réduire ou d'éliminer complètement le bruit. Le résultat de certains d'entre eux donne une meilleure interprétation visuelle ou une meilleure atténuation, et certains d'entre eux donnent un excellent lissage du bruit, et un autre type donne des bords et une forme à l'image.

Le filtrage dépend de deux choses : l'intensité du bruit et la taille du noyau, et le degré de lissage est proportionnel à la taille du noyau. Plus la taille du noyau est grande, plus le lissage est important. C'est pourquoi nous remarquons le flou de l'image dans les filtres utilisés avec un gros noyau. On peut noter que le principal inconvénient de ces filtres est qu'ils entraînent une perte d'information dans les zones proches des lignes et des bords.

Bibliographie

[1] Wikipedia : https://fr.wikipedia.org/wiki/Traitement_d%27images, consulter le : 04/04/2022

[2] SANDELI Mohamed. Traitement d'images par des approches bio-inspirées Application à la segmentation d'images.Université Constantine 2. 2014.

[3] CHIKH Mohammed Tahar. Amélioration des images par un modèle de réseau de neurones (Comparaison avec les filtres de base).Université Abou-Bakr Belkaid – Tlemcen.2011.

[4] MERABET Nabila et MAHLIA Meriem. Recherche d'images par le contenu.Université Abou-Bakr Belkaid –Tlemcen.2011.

[5] HOUASSINE Charif. Segmentation d'images par une approche biomimétique hybride.Université M'hamed Bougara-Boumerdes. 2012.

[6] SLIME Samir. Environnement de segmentation d'image à base d'une approche biomimétique. E.N.I.Algérie. 2007/2008.

[7] LAKHDARI Mohamed. Segmentation d'images par contour actif en appliquant les algorithmes génétiques. I.N.I.Algérie .Juin 2008.

[8] DUPUPET Maxence², Caractéristiques de l'image numérique. DUPUPET Maxence, L'image numérique Générez traitez et exploitez vos images.ENI : SOBIMA. Novembre 2019.

[9] ALLEAU Christophe. Transmettre et stocker de l'information.Univ Européenne.2018

[10] K. Aounallah, les approches de segmentation d'image par coopération régions contours. 2010.

[11] Sukhjinder Kaur, Noise Types and Various Removal Techniques. Volume 04. February 2015.

[12] M. M. Puranik and S.V.Halse, Study of Various Types of Noises in Digital Images. Volume 57 .March 2018.

[13] LAMRAOUI Djedjiga et SLIMANI Hanane. Filtrage des images par différentes approches. Université M'hamed Bougara-Boumerdes. 2017.

[14] Google : https://sites.google.com/site/androidtraitementimage/traitement-d-images, consulter le : 07/05/2022

[15] Maïtine Bergounioux, Quelques méthodes de filtrage en Traitement d'Image, Cours donné dans le cadre d'une école CIMPA, 24 Jan 2011.

[16] u-strasbg : https://dpt-info.u-strasbg.fr/~cronse/TIDOC/FILTER/lirelin.html, consulter le : 04/04/2022

[17] https://www.pfl-cepia.inra.fr/index.php?page=tutoImg-erosion-dilatation, consulter le : 04/04/2022

[18] esiee : https://perso.esiee.fr/~perretb/I5FM/TAI/morpho/index.html#, consulter le : 10/04/2022

[19] stringfixer : https://stringfixer.com/fr/Bilateral_filter, consulter le : 04/04/2022

[20] Wikipedia : https://fr.wikipedia.org/wiki/Filtre_de_Canny, consulter le : 07/05/2022

[21] HADJ ABDELKADER Kheir eddine et BOUNOUAHINE Belhadj. Réduction du bruit des images par filtrage spatial. Université Saad Dahleb - Blida. 2008.

[22] http://morpheo.inrialpes.fr/people/Boyer/Teaching/M2PGI/c3.pdf,consulter le : 12/05/2022

[23] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/,consulter le : 04/04/2022

[24] Geeksforgeeks : https://www.geeksforgeeks.org/libraries-in-python/,consulter le : 04/04/2022

[25] Developpez : https://xphilipp.developpez.com/articles/filtres/?page=page_3#LIII, consulter le : 10/04/2022