

UNIVERSITÉ CLAUDE BERNARD LYON1  
FACULTÉ DES SCIENCES ET TECHNOLOGIES  
DÉPARTEMENT INFORMATIQUE



**Traitement d'image**

---

**Rapport de TP traitement d'image**

---

*Présenté par :*

*SAHNOUNE Yacine  
LOUAIFI Azouaou*

*Encadré par :*

*Mme SAIDA. BOUAKAZ*

**Année universitaire 2024-2025**

# Table des matières

1	Introduction . . . . .	1
2	Premier pas . . . . .	1
3	Filtre médian . . . . .	2
4	La convolution . . . . .	4
4.1	Filtrage moyenneur . . . . .	4
5	Filtrage Gaussien . . . . .	6
5.1	Filtrage Gaussien : Génération et Application du noyau . . . . .	6
6	Filtre différentiel de Sobel et normalisation . . . . .	8
7	Filtre Laplacien et normalisation . . . . .	10
8	Transformation d'histogramme . . . . .	12
8.1	Calcul et affichage de l'histogramme . . . . .	12
8.2	Calcul et affichage de l'histogramme cumulé . . . . .	13
8.3	La Look-Up Table (LUT) . . . . .	14
8.4	Égalisation d'histogramme . . . . .	16
8.5	Étirement d'histogramme . . . . .	17
9	Implémentation d'une console interactive et fonctionnalités bonus . . . . .	21
10	Conclusion . . . . .	21

# 1 Introduction

Ce TP porte sur la compréhension et la mise en pratique des concepts fondamentaux du traitement d'images. Il aborde des notions clés telles que les pixels, les niveaux de gris, les histogrammes et l'opération de convolution. L'objectif principal est de traduire ces concepts théoriques en implémentations concrètes à travers des algorithmes en C++.

Dans ce cadre, nous devons réaliser plusieurs algorithmes de base pour le traitement d'images :

- **Le filtrage** pour lisser les images et réduire le bruit ;
- **La convolution** en utilisant différents noyaux pour appliquer divers effets ;
- **Les transformations d'histogramme** pour visualiser les niveaux de gris, réajuster le contraste ou transformer l'intensité lumineuse d'une image.

Ce travail permettra de démontrer notre compréhension des principes fondamentaux du traitement d'images.

Le code source de ce projet est accessible sur La Forge à l'adresse suivante : <https://forge.univ-lyon1.fr/p2413794/traitement-images.git>.

## 2 Premier pas

Pour commencer, il nous a été recommandé de bien comprendre le fonctionnement de la bibliothèque **OpenCV**, qui offre des fonctionnalités très intéressantes dans le domaine du traitement d'images. OpenCV permet notamment la manipulation des pixels et la gestion des matrices associées aux images.

La première étape consistait à apprendre comment manipuler une image à l'aide d'un code C++. L'idée était de comprendre qu'une image peut être représentée comme une matrice de pixels, de récupérer cette matrice et de la stocker dans une matrice en C++. Après traitement, le résultat devait être une nouvelle matrice de pixels qu'il fallait reconvertir en image.

Cependant, nous avons rencontré une contrainte lors de la transformation d'une image en matrice : chaque pixel est initialisé avec trois composantes de couleur **BGR** (Bleu, Vert, Rouge). Cette structure rend les calculs complexes, car chaque composante doit être traitée indépendamment. Pour résoudre ce problème, nous avons mis en place les méthodes suivantes :

- `std::vector<std::vector<int>> convertImageToVector` : permet de convertir une image en une matrice de pixels.
- `cv::Mat convertVectorToImage` : permet de reconvertir une matrice de pixels en image.
- `rgbToInt` : permet de convertir les trois composantes BGR d'un pixel en une seule valeur entière sur **24 bits**. Pour cela, nous avons assigné les **8 bits de poids faible** à la couleur bleue, les **8 bits du milieu** à la couleur verte, et les **8 bits de poids fort** à la couleur rouge.
- `intToRgb` : réalise l'opération inverse en décomposant une valeur entière sur 24 bits pour extraire les trois composantes BGR d'un pixel.

Ces méthodes ont permis de simplifier considérablement les calculs et la manipulation des pixels dans nos algorithmes de traitement d'images.

### 3 Filtre médian

Le **filtre médian** est une méthode utilisée dans le traitement d'images pour **réduire le bruit** tout en **préservant les contours** des objets présents dans une image. Il est particulièrement efficace contre le bruit impulsionnel, aussi appelé *bruit "sel et poivre"*.

#### Principe du filtre médian

**Structure des images :** Une image numérique est représentée sous forme d'une **matrice de pixels**. Chaque pixel contient des informations de couleur composées de trois canaux : **BGR** (Bleu, Vert, Rouge).

**Étapes du filtrage :** Pour appliquer le filtre médian sur une image :

- **Sélection d'une fenêtre de pixels :** Une fenêtre carrée de taille définie (ex : 3x3 ou 5x5) est centrée sur le pixel courant.
- **Extraction des valeurs des voisins :** Les valeurs des pixels situés dans la fenêtre sont extraites pour chaque composante de couleur : **B**, **G** et **R**.
- **Tri des valeurs :** Les valeurs des voisins pour chaque canal (**B**, **G** et **R**) sont triées de manière ascendante.
- **Calcul de la médiane :** La valeur médiane (celle située au milieu dans la liste triée) est sélectionnée pour chaque canal.
- **Recomposition du pixel :** Les trois médianes obtenues pour les canaux **B**, **G** et **R** sont recombinées pour former le nouveau pixel.

**Parcours de l'image :** Le processus est répété pour chaque pixel de l'image en parcourant l'ensemble des lignes et colonnes.

**Gestion des bords :** Lorsque la fenêtre dépasse les limites de l'image (sur les bords), seules les valeurs des pixels valides sont utilisées pour le calcul.



Image sans filtre



Image OpenCV



Image avec filtre médian

## 4 La convolution

La **convolution** est une opération mathématique fondamentale dans le traitement d'images. Elle consiste à appliquer un **noyau** (ou filtre), qui est une petite matrice, à chaque pixel de l'image et ses voisins pour calculer une nouvelle valeur de pixel.

Cette opération permet d'effectuer divers traitements comme le **lissage**, la **détection de contours** ou encore l'application d'effets spéciaux. La convolution repose sur un produit élément par élément entre le **noyau** et les pixels voisins, suivi d'une somme des résultats.

Dans les sections suivantes, j'utiliserai cette technique pour implémenter des filtres qui nécessitent un **noyau** spécifique.

### 4.1 Filtrage moyenneur

Le **filtrage moyenneur**, également appelé **filtre passe-bas**, est une technique de traitement d'image qui permet de **lisser une image** en réduisant les variations locales de luminosité. Cette méthode est particulièrement efficace pour atténuer le bruit dans une image tout en la rendant plus homogène.

**Principe de fonctionnement de notre implémentation :** Le filtrage moyenneur utilise un **noyau** (matrice) de taille  $L \times L$  rempli de valeurs égales à  $\frac{1}{L \times L}$ . Ce noyau est appliqué à chaque pixel de l'image ainsi qu'à ses voisins. L'algorithme effectue les étapes suivantes :

- **Parcours de l'image :** Chaque pixel de l'image est visité.
- **Application du noyau :** Le noyau est centré sur le pixel courant et multiplie les valeurs des pixels voisins par les coefficients du noyau.
- **Somme des valeurs pondérées :** Les produits obtenus sont additionnés pour calculer une nouvelle valeur moyenne du pixel.
- **Remplacement du pixel :** La nouvelle valeur est assignée au pixel courant dans l'image filtrée.

**Convolution et noyau :** Le noyau utilisé dans le filtrage moyenneur est une matrice uniforme où chaque coefficient a la même valeur. La **convolution** de ce noyau avec l'image revient à effectuer une moyenne pondérée des pixels voisins. La taille  $L \times L$  du noyau détermine l'ampleur du lissage :

- Une petite taille de noyau (ex :  $3 \times 3$ ) lisse légèrement l'image.
- Une grande taille de noyau (ex :  $7 \times 7$ ) produit un lissage plus important, mais au détriment des détails.

**Conclusion :** Ce filtrage est un exemple d'opération de **convolution** appliquée à une image. Il est simple à implémenter et efficace pour réduire le bruit tout en conservant les contours globaux. Le noyau moyenneur sera utilisé comme base dans les prochains filtres.



Image sans filtre



Image OpenCV



Image avec filtre moyenneur

## 5 Filtrage Gaussien

### 5.1 Filtrage Gaussien : Génération et Application du noyau

Le **filtre Gaussien** est un filtre de lissage qui permet de réduire le bruit dans une image tout en conservant des transitions douces. Il utilise un **noyau Gaussien** généré à partir de la formule mathématique suivante :

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-x^2 - y^2}{2\sigma^2}\right)$$

où  $x$  et  $y$  représentent les positions relatives des pixels voisins par rapport au centre du noyau, et  $\sigma$  (sigma) contrôle l'étalement de la fonction Gaussienne. Plus la valeur de  $\sigma$  est grande, plus le flou appliqué sera important.

**Génération du noyau Gaussien** Le noyau Gaussien est une matrice carrée de taille  $L \times L$  où chaque coefficient est calculé en utilisant la formule ci-dessus. Voici les étapes pour générer un noyau Gaussien :

- **Initialisation du noyau** : Une matrice vide est créée avec la taille  $L \times L$ . La taille  $L$  doit être impaire pour permettre un centrage correct du noyau.
- **Calcul des valeurs** : Chaque coefficient  $G(x, y)$  est calculé en fonction des coordonnées  $x$  et  $y$  par rapport au centre du noyau. Les valeurs sont pondérées par la fonction exponentielle pour donner plus d'importance aux pixels proches du centre.
- **Normalisation** : Après le calcul, le noyau est normalisé pour que la somme de tous ses coefficients soit égale à 1. Cela permet de conserver la luminosité globale de l'image après l'application du filtre.

**Application du noyau Gaussien** : L'application du noyau Gaussien repose sur une opération de **convolution** entre le noyau et l'image. Pour chaque pixel de l'image :

- Le noyau est centré sur le pixel courant.
- Chaque coefficient du noyau est multiplié par la valeur du pixel correspondant dans l'image (et ses voisins).
- Les résultats sont additionnés pour obtenir une nouvelle valeur pondérée pour le pixel courant.

L'image est traitée canal par canal (Bleu, Vert et Rouge), ce qui permet d'appliquer le filtre indépendamment sur chaque composante.

**Deux modes d'utilisation** : Deux fonctions sont développées pour appliquer le filtrage Gaussien :

1. **Filtrage Gaussien avec choix de  $\sigma$**  : L'utilisateur peut spécifier la taille du noyau ( $L \times L$ ) et la valeur de  $\sigma$ . Le noyau est alors généré dynamiquement en utilisant la formule Gaussienne.
2. **Filtrage Gaussien avec un noyau pré-généré** : Cette méthode permet de fournir directement un noyau Gaussien déjà calculé. Cela évite de recalculer le noyau, ce qui peut être avantageux pour des raisons de performance.

**Conclusion :** Le filtre Gaussien est particulièrement utile pour le prétraitement des images dans des tâches telles que la réduction de bruit ou la préparation d'une image pour la détection de contours. Grâce à son noyau paramétrable, il offre une grande flexibilité dans le contrôle du niveau de lissage.



Image sans filtre



Image OpenCV



Image avec filtre Gaussien

## 6 Filtre différentiel de Sobel et normalisation

Le **filtre de Sobel** est une méthode classique utilisée pour la **détection de contours** dans une image. Il permet de calculer les gradients d'intensité dans deux directions principales : horizontale et verticale.

**Principe de fonctionnement de notre implémentation :** Le filtre Sobel repose sur la convolution de l'image avec deux **noyaux Sobel** :

- Le noyau  $G_x$  pour les gradients horizontaux.
- Le noyau  $G_y$  pour les gradients verticaux.

Pour chaque pixel de l'image :

1. **Convolution avec  $G_x$  et  $G_y$**  : Les deux noyaux sont appliqués séparément pour calculer les composantes horizontale et verticale des gradients pour chaque canal de couleur (**B**, **G**, **R**).
2. **Magnitude du gradient** : La magnitude est calculée pour chaque canal à l'aide de la formule :

$$\text{magnitude} = \sqrt{(G_x)^2 + (G_y)^2}$$

La valeur globale de l'intensité est ensuite obtenue en moyennant les magnitudes des trois canaux :

$$\text{intensité} = \frac{\text{magnitude}_B + \text{magnitude}_G + \text{magnitude}_R}{3}$$

3. **Normalisation et seuillage** : Une normalisation est appliquée pour ramener les valeurs dans la plage [0, 255] :

$$\text{normalizedIntensity} = \min \left( 255, \frac{\text{intensité}}{255} \times 255 \right)$$

Ensuite, un **seuil** est appliqué pour éliminer les variations faibles dues au bruit :

Si  $\text{intensité} > 30$ , alors le pixel est conservé, sinon il est mis à 0.

**Importance de la normalisation** : La normalisation joue un rôle clé pour garantir des résultats exploitables :

- **Échelle cohérente** : Les valeurs résultant de la convolution peuvent dépasser 255, ce qui rend leur affichage incorrect sans normalisation.
- **Visibilité des contours** : En ramenant les valeurs à une échelle uniforme, les bords détectés apparaissent clairement, avec une intensité maximale pour les contours forts.
- **Réduction du bruit** : Le seuillage permet d'éliminer les détails insignifiants et les variations faibles dues au bruit.

**Conclusion** : Le filtre de Sobel est une méthode efficace pour détecter les contours d'une image. Grâce à la **normalisation** et au **seuillage**, il est possible de mettre en évidence les bords les plus marqués tout en réduisant l'impact des artefacts mineurs.



Image sans filtre



Image OpenCV

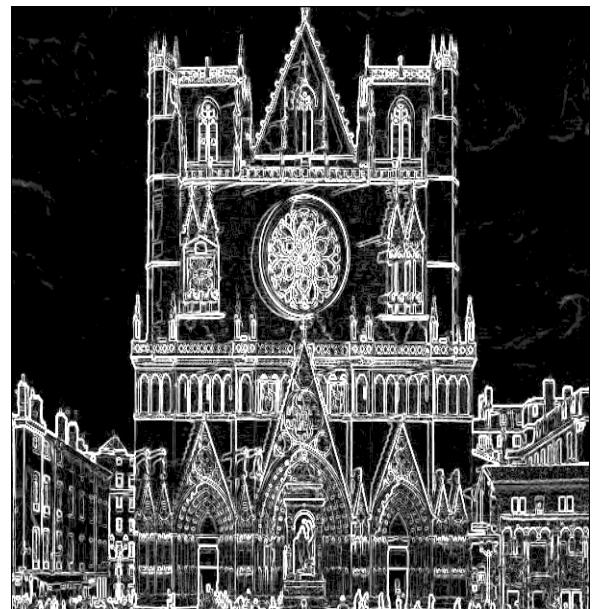


Image avec filtre Sobel

## 7 Filtre Laplacien et normalisation

Le **filtre Laplacien** est un filtre différentiel utilisé pour détecter les contours dans une image en calculant les variations d'intensité dans toutes les directions. Contrairement au filtre de Sobel qui calcule les gradients horizontaux et verticaux séparément, le Laplacien utilise un seul noyau pour analyser la variation locale autour d'un pixel.

**Principe de fonctionnement de notre implémentation :**

1. **Application du noyau Laplacien :** Le noyau Laplacien est appliqué sur chaque pixel de l'image et ses voisins en effectuant une **convolution**. Chaque coefficient du noyau est multiplié par les valeurs des pixels correspondants, et la somme pondérée est calculée pour chaque canal de couleur (**B**, **G**, **R**).
2. **Calcul de l'intensité des contours :** Les contributions des trois canaux (**B**, **G**, **R**) sont additionnées pour calculer une **intensité globale** du contour. Cette intensité est définie comme suit :

$$\text{intensité} = \frac{|\text{sumBGR}_B| + |\text{sumBGR}_G| + |\text{sumBGR}_R|}{3}$$

où  $\text{sumBGR}$  représente la somme pondérée obtenue après convolution pour chaque canal.

3. **Normalisation des valeurs positives :** Seules les valeurs positives des intensités sont conservées et normalisées dans la plage [0, 255] en utilisant la formule :

$$\text{normalizedIntensity} = \min \left( 255, \frac{\text{intensité}}{255} \times 255 \right)$$

Cela garantit que les valeurs ne dépassent pas 255 et peuvent être affichées correctement dans une image en niveaux de gris.

4. **Seuillage :** Un seuil est appliqué pour éliminer les faibles intensités (non-contours) qui pourraient être dues au bruit. Si l'intensité calculée est inférieure à un seuil (par exemple, 30), le pixel est mis à noir :

$$\text{Si } \text{intensité} \leq 30, \text{ alors } \text{pixel} = 0.$$

Cela permet de conserver uniquement les contours les plus marqués.

**Normalisation dans le contexte du Laplacien :** La normalisation ici est appliquée sur les **valeurs positives** des intensités obtenues après la convolution. Les valeurs négatives, qui peuvent apparaître en raison des propriétés du noyau Laplacien, sont mises à zéro car elles ne sont pas exploitables pour l'affichage des contours.

**Conclusion :** Le filtre Laplacien est particulièrement efficace pour détecter les contours grâce à sa capacité à identifier les variations d'intensité dans toutes les directions. La normalisation des intensités positives, suivie du seuillage, permet de produire une image en niveaux de gris où seuls les contours les plus significatifs sont visibles. Cette méthode offre un bon compromis entre précision et simplicité.

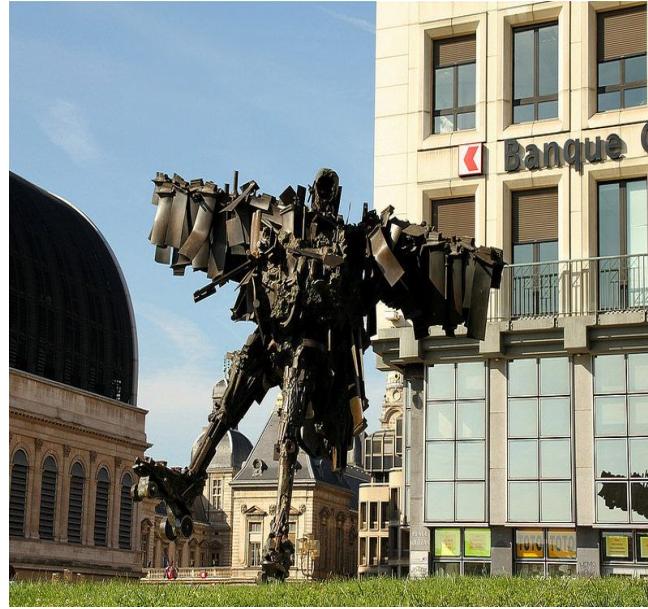


Image sans filtre

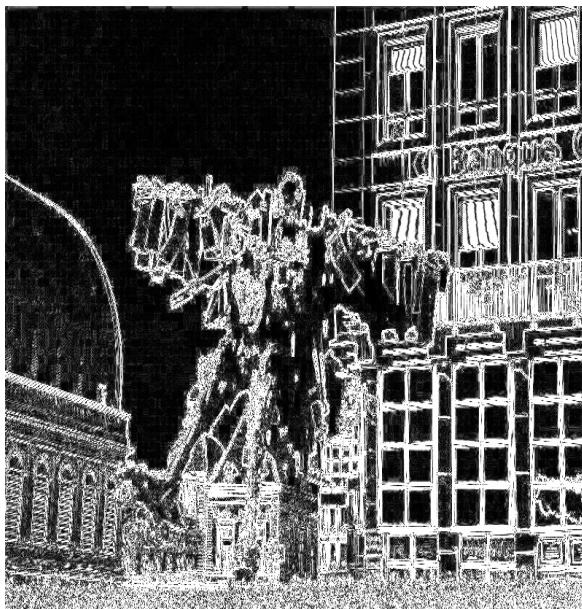


Image OpenCV

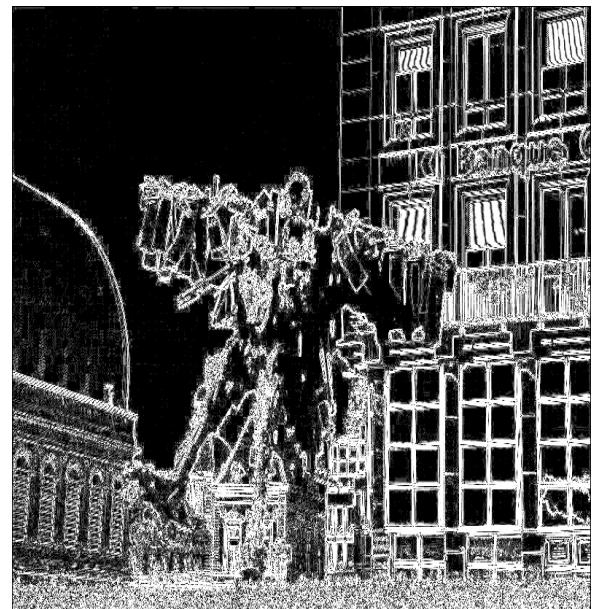


Image avec filtre Laplacien

# 8 Transformation d'histogramme

## 8.1 Calcul et affichage de l'histogramme

Le calcul de l'**histogramme** est une étape importante dans l'analyse d'une image. Dans ce cas, l'image est convertie en **niveaux de gris** afin de simplifier l'analyse en ne tenant compte que des intensités lumineuses.

**Principe de fonctionnement :**

1. **Conversion en niveaux de gris** : L'image est traitée en niveaux de gris où chaque pixel possède une unique valeur d'intensité comprise entre 0 (noir) et 255 (blanc). Cela permet d'éliminer les informations inutiles des trois canaux de couleur (**BGR**).
2. **Calcul de l'histogramme** : L'histogramme est construit en comptant le nombre de pixels pour chaque niveau d'intensité. Les étapes sont :
  - Parcourir chaque pixel de l'image.
  - Récupérer l'intensité du pixel (intensity) directement en niveaux de gris.
  - Incrémenter un tableau de 256 valeurs où chaque indice correspond à un niveau d'intensité.

Mathématiquement, l'histogramme  $H$  est défini par :

$$H[i] = \text{nombre de pixels de l'image ayant une intensité } i, \text{ où } i \in [0, 255].$$

**Normalisation pour l'affichage** : Afin d'afficher l'histogramme sous forme de graphique, les valeurs de l'histogramme sont **normalisées** pour s'adapter à la taille d'une fenêtre graphique. La formule utilisée est la suivante :

$$H_{\text{norm}}[i] = \text{round} \left( \frac{H[i] \times \text{Hauteur}}{\text{Valeur maximale}} \right)$$

où :

- $H[i]$  est la fréquence d'intensité  $i$ .
- Hauteur est la taille maximale de l'histogramme sur l'axe Y.
- Valeur maximale est le plus grand nombre de pixels pour une intensité donnée.

**Affichage de l'histogramme** : L'histogramme est dessiné sous forme de barres verticales où :

- L'axe horizontal (X) représente les **niveaux d'intensité** de 0 à 255.
- L'axe vertical (Y) représente la **fréquence** des pixels pour chaque intensité.

Une grille est ajoutée pour faciliter la lecture des valeurs sur les axes. Chaque barre de l'histogramme est remplie en **noir** pour mieux représenter les fréquences des niveaux de gris.

**Simplification en niveaux de gris :** Le choix de travailler en niveaux de gris plutôt qu'en couleur (**BGR**) présente plusieurs avantages :

- Réduction de la complexité : un seul canal d'intensité est analysé.
- Simplification du traitement : les informations de couleur ne sont pas nécessaires pour le calcul de l'histogramme.
- Rapidité : le parcours de l'image est optimisé puisque seul le canal d'intensité est utilisé.

**Conclusion :** L'histogramme en niveaux de gris permet une représentation simple et efficace de la répartition des intensités lumineuses dans une image. La normalisation assure un affichage clair et lisible, tout en éliminant les détails superflus liés aux couleurs.

## 8.2 Calcul et affichage de l'histogramme cumulé

L'**histogramme cumulé** est une extension de l'histogramme classique. Il permet de représenter la distribution accumulée des intensités dans une image. Cet outil est particulièrement utile pour l'analyse de la répartition des pixels et pour des traitements comme l'**égalisation d'histogramme**.

**Calcul de l'histogramme cumulé :** L'histogramme cumulé est obtenu en additionnant, pour chaque niveau d'intensité  $i$ , toutes les fréquences des intensités précédentes. La formule de l'histogramme cumulé  $H_c$  est donnée par :

$$H_c[i] = \sum_{k=0}^i H[k], \quad \text{où } i \in [0, 255]$$

où  $H[k]$  est l'histogramme des niveaux de gris calculé précédemment.

- $H_c[0]$  est initialisé avec  $H[0]$ .
- Pour chaque intensité  $i$ , on additionne  $H_c[i - 1]$  et  $H[i]$ .

**Normalisation pour l'affichage :** Afin de représenter l'histogramme cumulé de manière visuelle, les valeurs de l'histogramme sont **normalisées** pour s'adapter à la taille d'une fenêtre graphique. La normalisation est réalisée selon la relation suivante :

$$H_{\text{cumul,norm}}[i] = \text{round} \left( \frac{H_c[i] \times \text{Hauteur}}{\text{Valeur maximale}} \right)$$

où :

- $H_c[i]$  est la valeur cumulée de l'histogramme à l'intensité  $i$ .
- Hauteur est la taille maximale de l'axe Y pour le graphique.
- Valeur maximale est la plus grande valeur dans l'histogramme cumulé.

**Affichage de l'histogramme cumulé :** L'histogramme cumulé est représenté sous forme de barres verticales où :

- **L'axe horizontal (X)** représente les niveaux d'intensité de 0 à 255.
- **L'axe vertical (Y)** représente la fréquence cumulée des intensités.

Les étapes d'affichage sont les suivantes :

1. Créer une image blanche pour l'histogramme cumulé.
2. Normaliser les valeurs de l'histogramme cumulé pour qu'elles s'adaptent à l'échelle choisie.
3. Dessiner les barres verticales en utilisant les valeurs normalisées.
4. Ajouter les **axes** et les **étiquettes** pour faciliter la lecture des niveaux d'intensité et des fréquences cumulées.

**Sauvegarde et affichage :** L'histogramme cumulé est affiché à l'écran et peut également être **sauvegardé** sous forme d'image pour une utilisation ultérieure :

Image sauvegardée : Images/histogramme\_cumule.png

**Utilité de l'histogramme cumulé :** L'histogramme cumulé est utilisé dans plusieurs applications :

- **Égalisation d'histogramme** : Améliorer le contraste d'une image.
- **Analyse de la répartition des pixels** : Comprendre si une image est trop sombre ou trop claire.
- **Seuil adaptatif** : Déterminer des seuils pour la segmentation d'image.

**Conclusion :** L'histogramme cumulé est un outil puissant pour visualiser et analyser la distribution globale des intensités d'une image. Sa normalisation permet un affichage clair et proportionné, facilitant ainsi son interprétation.

### 8.3 La Look-Up Table (LUT)

La **LUT** (Look-Up Table) est une structure utilisée pour transformer les valeurs des pixels d'une image de manière rapide et efficace. Dans le contexte de ton code, la LUT est créée à partir de l'**histogramme cumulé** pour appliquer une transformation sur les niveaux d'intensité.

**Principe de la LUT :** La LUT est un tableau de taille fixe (256 pour une image en niveaux de gris) dans lequel chaque entrée correspond à un niveau d'intensité d'entrée  $i$  et fournit un niveau de sortie  $LUT[i]$ . Cette table est construite à partir des valeurs de l'**histogramme cumulé normalisé**.

L'idée générale est de transformer l'image en utilisant la relation suivante :

$$LUT[i] = \text{valeur normalisée de l'histogramme cumulé}$$

Cette transformation est souvent utilisée pour l'**égalisation d'histogramme**, afin d'améliorer la répartition des niveaux d'intensité dans l'image.

**Construction de la LUT :** Le code construit la LUT en normalisant les valeurs de l'histogramme cumulé pour les ramener dans la plage [0, 255]. La formule utilisée est la suivante :

$$\text{LUT}[i] = \text{round} \left( \frac{\text{Histogramme cumulé}[i] \times 255}{\text{TotalPixels}} \right)$$

où :

- Histogramme cumulé[i] est la valeur cumulée pour l'intensité  $i$ .
- TotalPixels est le nombre total de pixels dans l'image.

**Normalisation avec  $C_{\min}$  :** Une autre méthode de création de la LUT, utilisée dans la fonction `creerLUT`, consiste à prendre en compte la première valeur non nulle de l'histogramme cumulé ( $C_{\min}$ ) pour améliorer la précision de la transformation. La formule devient alors :

$$\text{LUT}[i] = \text{round} \left( \frac{\text{Histogramme cumulé}[i] - C_{\min}}{\text{TotalPixels} - C_{\min}} \times 255 \right)$$

où :

- $C_{\min}$  est la plus petite valeur non nulle dans l'histogramme cumulé.
  - Cette méthode permet d'éliminer les biais liés aux intensités initialement absentes.
- 

**Affichage de la LUT :** La LUT est représentée sous forme de graphe où chaque niveau d'intensité d'entrée est associé à son niveau d'intensité de sortie. Dans ton code :

- L'axe horizontal représente les **niveaux d'intensité d'entrée** allant de 0 à 255.
  - L'axe vertical représente les **niveaux d'intensité de sortie** après transformation par la LUT.
  - Une ligne rouge est tracée pour visualiser la relation entre l'entrée et la sortie.
- 

**Utilité de la LUT :** La LUT permet d'appliquer une transformation sur les pixels de manière très rapide car elle remplace les calculs complexes par de simples opérations d'accès au tableau. Elle est utilisée dans plusieurs contextes :

- **Égalisation d'histogramme** : Amélioration du contraste en redistribuant les niveaux d'intensité.
  - **Correction gamma** : Ajustement de la luminosité des images.
  - **Transformations personnalisées** : Modification des niveaux d'intensité selon des besoins spécifiques.
- 

**Conclusion :** La LUT est une technique efficace pour transformer rapidement les niveaux d'intensité d'une image. Dans ton code, elle est construite à partir de l'histogramme cumulé pour assurer une transformation normalisée des intensités, permettant ainsi une amélioration du contraste ou une égalisation précise.

## 8.4 Égalisation d'histogramme

L'**égalisation d'histogramme** est une technique de traitement d'image permettant d'améliorer le contraste d'une image en redistribuant les niveaux d'intensité. Elle utilise l'**histogramme cumulé** pour transformer les niveaux de gris de manière à obtenir une répartition plus homogène.

**Principe de fonctionnement :** L'égalisation d'histogramme consiste à appliquer une transformation sur chaque niveau d'intensité  $i$  d'une image pour obtenir un nouveau niveau d'intensité  $i'$ . La transformation est basée sur l'**histogramme cumulé normalisé** et est définie par :

$$i' = \text{LUT}[i] = \text{round} \left( \frac{\text{HistogrammeCumule}[i] \times 255}{\text{TotalPixels}} \right)$$

où :

- $\text{HistogrammeCumule}[i]$  est la valeur cumulée des intensités jusqu'à  $i$ .
  - $\text{TotalPixels}$  est le nombre total de pixels dans l'image.
  - La table LUT (*Look-Up Table*) stocke les nouvelles valeurs d'intensité pour chaque niveau d'entrée.
- 

**Étapes du code :** Le code effectue les étapes suivantes pour égaliser l'histogramme d'une image en niveaux de gris :

1. **Calcul de l'histogramme :** L'histogramme des niveaux de gris est calculé en parcourant chaque pixel de l'image et en comptant la fréquence de chaque intensité.
2. **Calcul de l'histogramme cumulé :** À partir de l'histogramme, l'histogramme cumulé est construit en additionnant les fréquences des intensités précédentes :

$$\text{HistogrammeCumule}[i] = \text{HistogrammeCumule}[i - 1] + \text{Histogramme}[i].$$

3. **Création de la LUT (Look-Up Table) :** La LUT est créée en normalisant les valeurs de l'histogramme cumulé pour les ramener dans la plage  $[0, 255]$  :

$$\text{LUT}[i] = \text{round} \left( \frac{\text{HistogrammeCumule}[i] \times 255}{\text{TotalPixels}} \right).$$

4. **Application de la transformation :** Pour chaque pixel de l'image, la nouvelle intensité est obtenue en utilisant la LUT :

$$\text{Nouveau niveau d'intensité} = \text{LUT}[\text{Ancien niveau d'intensité}].$$

Une nouvelle matrice est créée pour stocker les valeurs transformées.

5. **Conversion de la matrice en image OpenCV :** La matrice égalisée est convertie en une image OpenCV pour un affichage visuel.
-

**Amélioration du contraste :** L'égalisation d'histogramme améliore le contraste d'une image en étalant les niveaux d'intensité sur toute la plage possible [0, 255]. Cela permet d'accentuer les détails dans les régions sombres et claires.

---

**Conclusion :** L'égalisation d'histogramme est une méthode simple mais puissante pour améliorer la visibilité des détails d'une image. En utilisant l'histogramme cumulé et une table LUT, le code optimise efficacement la répartition des intensités pour produire une image au contraste amélioré.

## 8.5 Étirement d'histogramme

L'**étirement d'histogramme** est une méthode de traitement d'image visant à améliorer le contraste en élargissant la plage des niveaux d'intensité présents dans l'image. Contrairement à l'égalisation d'histogramme, qui redistribue les intensités pour uniformiser leur répartition, l'étirement ajuste directement les niveaux d'intensité en fonction des valeurs minimales et maximales observées.

---

**Principe de fonctionnement :** L'étirement d'histogramme repose sur l'utilisation d'une **LUT (Look-Up Table)** pour transformer les niveaux d'intensité des pixels. La transformation linéaire est définie comme suit :

$$\text{LUT}[i] = \text{round} \left( \frac{\text{HistogrammeCumule}[i] - C_{\min}}{\text{TotalPixels} - C_{\min}} \times 255 \right)$$

où :

- $\text{HistogrammeCumule}[i]$  est l'histogramme cumulé au niveau  $i$ .
- $C_{\min}$  est la première valeur non nulle dans l'histogramme cumulé.
- $\text{TotalPixels}$  est le nombre total de pixels dans l'image.
- 255 est la valeur maximale pour un pixel en niveaux de gris.

L'objectif est d'étirer les intensités existantes pour occuper toute la plage dynamique [0, 255].

---

**Étapes du code :** Le code réalise les étapes suivantes :

1. **Calcul de l'histogramme** : L'histogramme est construit pour obtenir les fréquences des niveaux d'intensité de l'image.
2. **Calcul de l'histogramme cumulé** : L'histogramme cumulé est calculé en additionnant les fréquences des intensités précédentes.
3. **Création de la LUT** : La LUT est générée en normalisant les valeurs cumulées pour les ajuster dans la plage dynamique [0, 255].
4. **Transformation des pixels** : Chaque pixel de l'image d'origine est transformé en utilisant la LUT :

$$\text{Nouvelle valeur} = \text{LUT}[\text{Ancienne valeur}].$$

Une nouvelle image est créée où les niveaux d'intensité ont été étirés.

5. **Résultat :** L'image finale possède un contraste amélioré grâce à l'étalement des niveaux de gris.

---

**Amélioration du contraste :** L'étirement d'histogramme améliore le contraste en maximisant la plage dynamique des niveaux d'intensité. Cela permet :

- d'accentuer les détails dans les zones sombres et claires,
  - de rendre l'image visuellement plus équilibrée et contrastée.
- 

**Conclusion :** L'étirement d'histogramme est une méthode efficace pour améliorer le contraste d'une image en ajustant linéairement les niveaux d'intensité. Grâce à l'utilisation de l'histogramme cumulé et d'une LUT, cette transformation est réalisée de manière rapide et optimisée.



Figure 6: Image de test

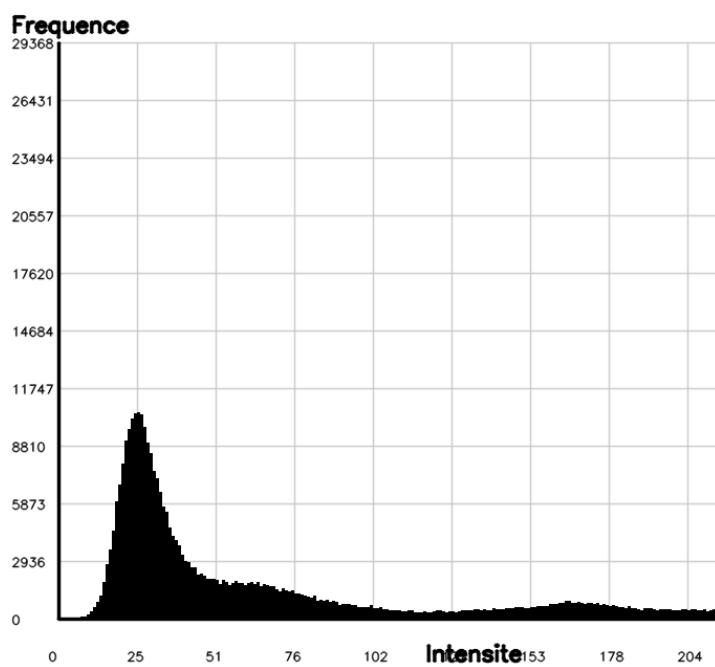


Figure 7: Histogramme

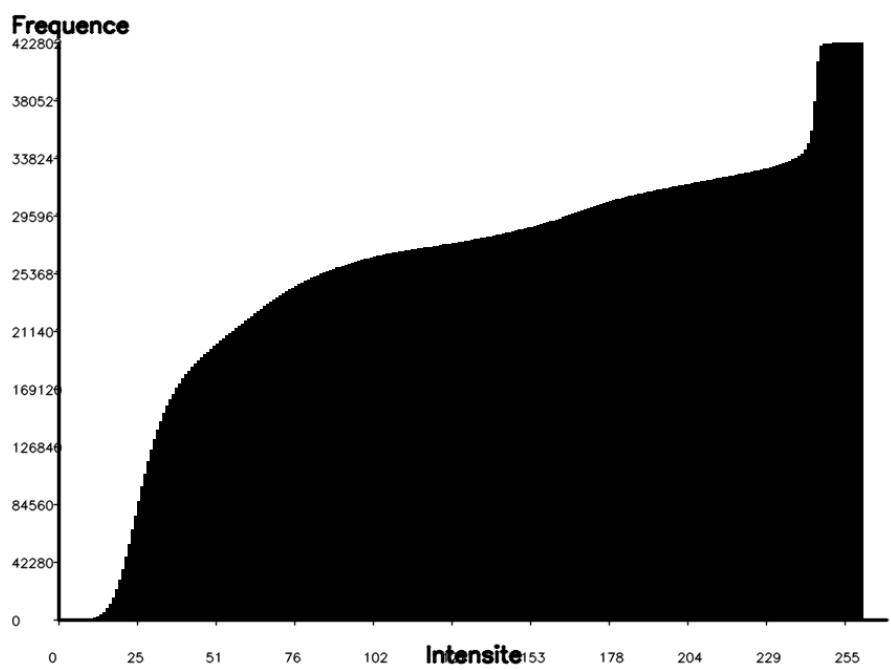


Figure 8: Histogramme cummulé



Figure 9: Image égalisée



Figure 10: Image étirée

## 9 Implémentation d'une console interactive et fonctionnalités bonus

Dans cette section, nous présentons une **console interactive** que nous avons intégrée dans notre programme. Cette console permet à l'utilisateur de choisir dynamiquement :

- **L'image de son choix** pour effectuer les traitements.
- **Le filtre à appliquer** parmi les différentes options disponibles.
- Les **noyaux de convolution** spécifiques associés à chaque filtre (par exemple, noyaux Sobel, Gaussien, Laplacien, etc.).

Cette approche interactive facilite les tests sur des images variées et permet à l'utilisateur d'expérimenter différentes configurations sans avoir à recompiler le code à chaque modification.

---

**Compilation et exécution du programme** : Pour exécuter le programme correctement, il est recommandé d'utiliser un environnement **Linux**. Si le programme ne s'exécute pas directement avec une commande de compilation classique en C++, nous invitons l'utilisateur à utiliser la commande suivante :

```
g++ -o <nom> <nomDuFichier.cpp> $(pkg-config --cflags --libs opencv4)
```

---

**Fonctionnalités bonus** : En plus des filtres de base implémentés, nous avons ajouté des fonctionnalités supplémentaires pour répondre à la **question bonus** :

- Tester avec plusieurs **noyaux de convolution** sur une image.
- Comparaison avec **OpenCv** sur une image.
- **Zoomer** sur une image.
- **Dézoomer** pour réduire l'affichage d'une image.
- **Redimensionner** une image selon les dimensions spécifiées par l'utilisateur.

Ces fonctionnalités permettent une manipulation plus avancée des images, offrant ainsi à l'utilisateur un contrôle interactif et personnalisé sur le traitement effectué.

---

## 10 Conclusion

En conclusion, nous avons pu réaliser différentes implémentations autour des deux parties principales : les **filtres** et les **histogrammes**. Cela nous a permis de mieux comprendre les bases du traitement d'image, notamment les principes de filtrage, de convolution, ainsi que les manipulations des niveaux d'intensité. Ces travaux nous ont offert une vision pratique et approfondie des techniques fondamentales dans ce domaine.