Université Paul Sabatier
Toulouse

Compte Rendu

Travaux pratiques de Traitement d'Images

M1 EEA

- Module KEAX7AK1 : Traitement des images
- Enseignant : A. Herbulot

- Réalisé par :
  - KABOU Abdeldjalil
  - IAICHE Achour Mehdi Anis

2024 / 2025

Travaux pratiques de Traitement d'Images

Objectif

Le but est manipuler des notions d'histogrammes et de filtrage pour des images en niveaux de gris.

1 Utilisation d'histogrammes

Explication des différences observées au niveau de l'histogramme et de l'image :

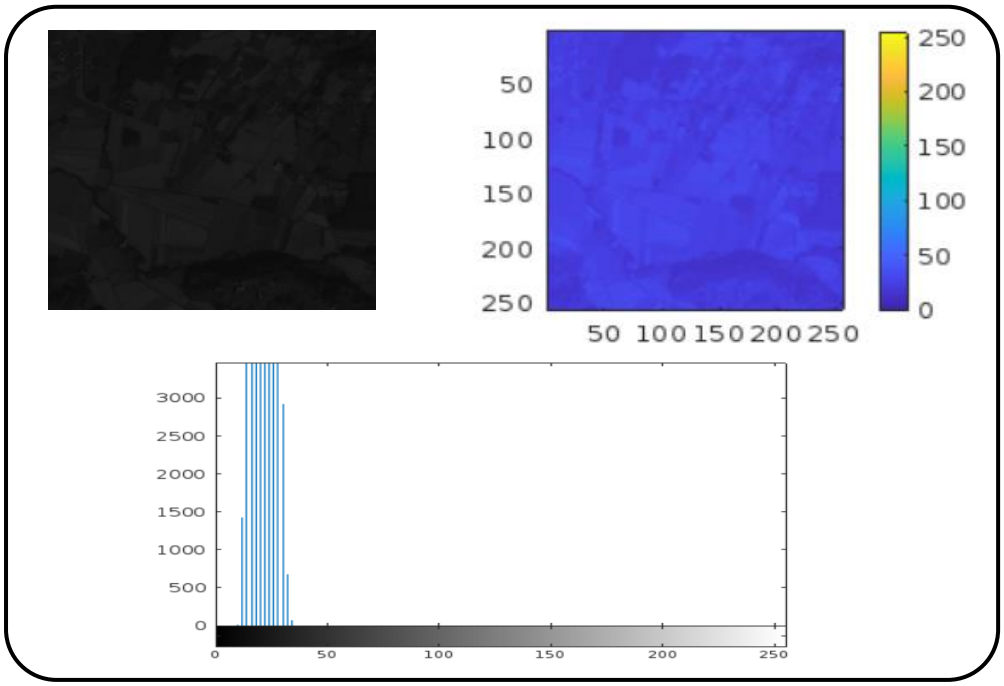


*Image '**aquitaine.png**' et son histogramme*

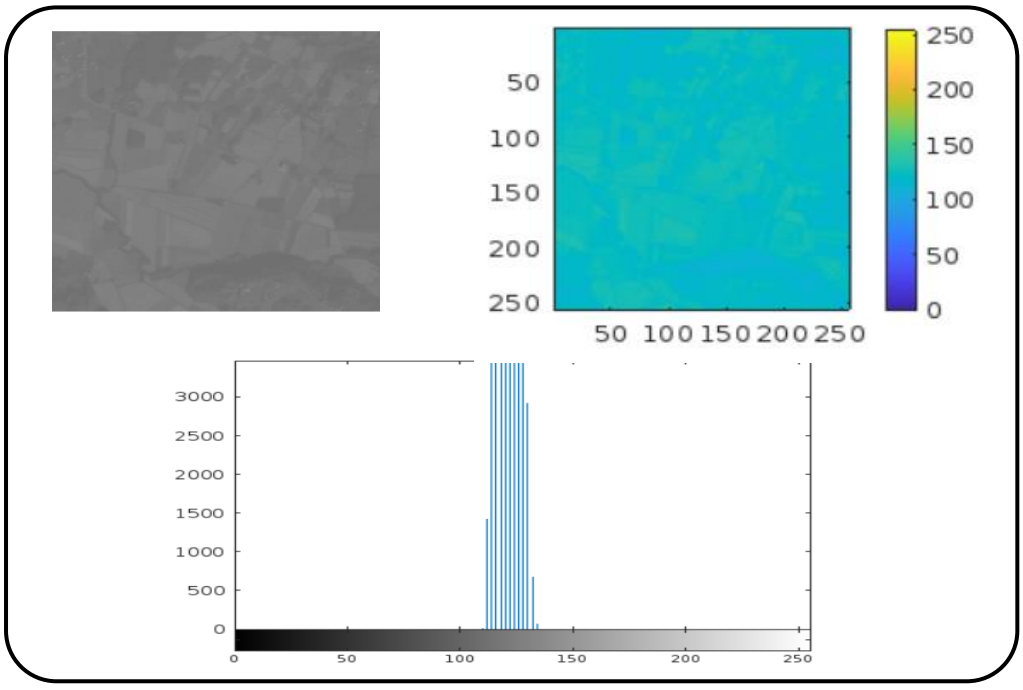


*Image **modifiée** en additionnant 100 à l'intensité de chaque pixel et son histogramme*

En ajoutant 100 à chaque pixel, on augmente l'intensité globale de l'image. Cette opération (ce décalage) rend l'image plus lumineuse.

- Au niveau de l'histogramme :

Dans l'histogramme modifiée, toutes valeurs de pixels sont déplacées vers la droite de 100 unités, car chaque intensité est augmentée de 100. Et pour éviter le débordement qui pourrait causer une saturation à 255, nous pouvons ajouter soit :

- La fonction min (image+100,255)
- Une condition, par exemple : *image (image > 255) = 255 ;*

- Au niveau de l'image :

L'image modifiée apparait globalement plus lumineuse que l'originale, avec une perte de détails dans les zones claires.

Explication des différences observées au niveau de l'histogramme et de l'image :

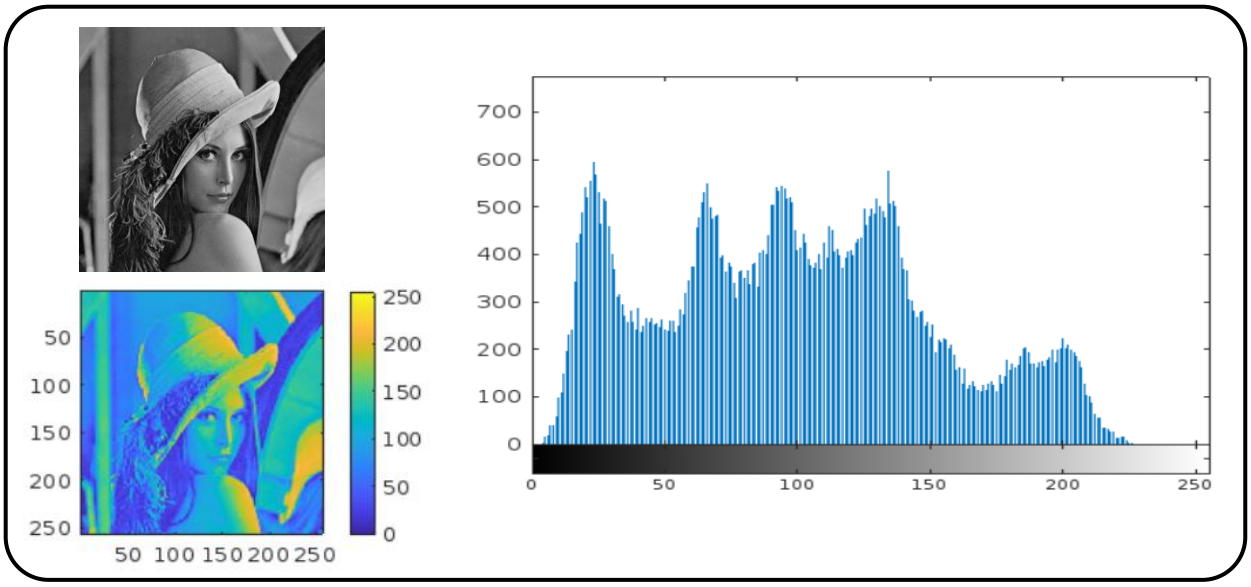


*Image **'lena.png'** et son histogramme*

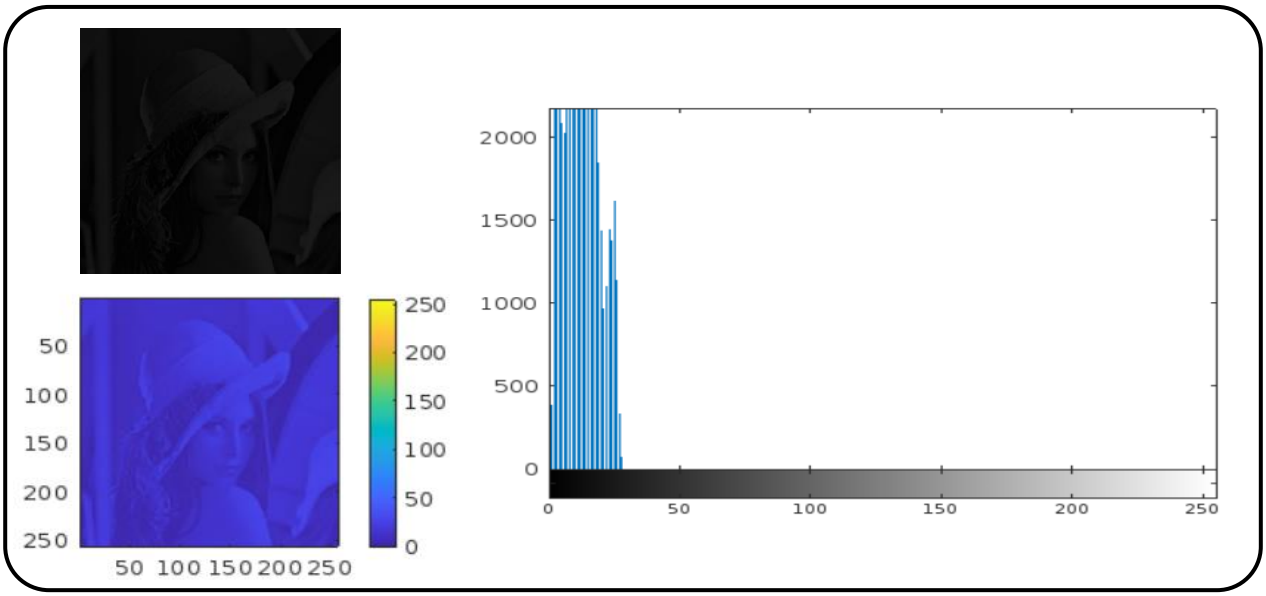


*Image **modifiée** en divisant par 8 l'intensité de chaque pixel et son histogramme*

Au contraire par apport la partie précédente, quand on divise chaque pixel par 8, les valeurs d'intensité sont réduites, ce qui assombrit l'image.

- Au niveau de l'histogramme :

L'histogramme de l'image modifiée montre que toutes les valeurs de pixels se déplacent vers la gauche (vers les intensités faibles), avec beaucoup de détails sont perdus dans la zone sombres.

- Au niveau de l'image :

L'image devient plus sombre, avec des détails moins visibles dans les zones initialement claires, car la dynamique est compressée.

Remarques :

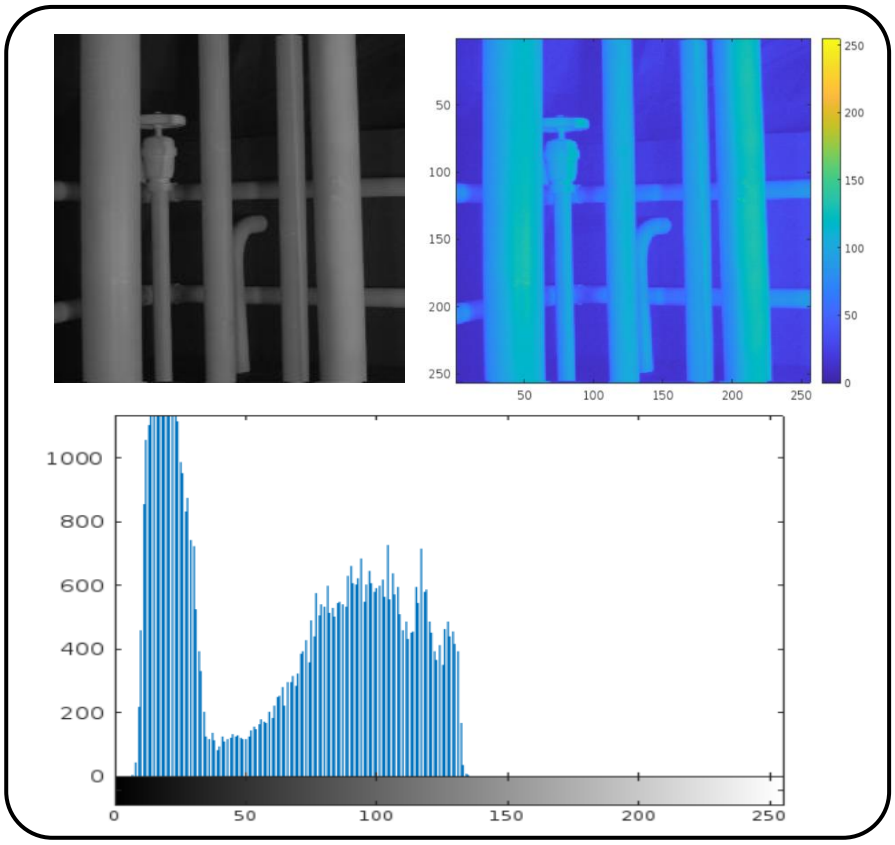


*Image '**micro1.png**' et son histogramme*

L'histogramme présente deux pics distincts ; l'un vers la gauche (intensités faibles, donc des pixels sombres), et un autre vers le milieu (intensités moyenne, donc des pixels gris).

Observation dans l'image :

- Image sombre avec des zones en gris moyen
- Absence de zones très claires ; puisque l'histogramme n'a pas de pics vers la droite (blancs)
- L'image a un contraste limité et se situé principalement dans des niveaux sombres et gris

Comparaison avant / après La Normalisation :

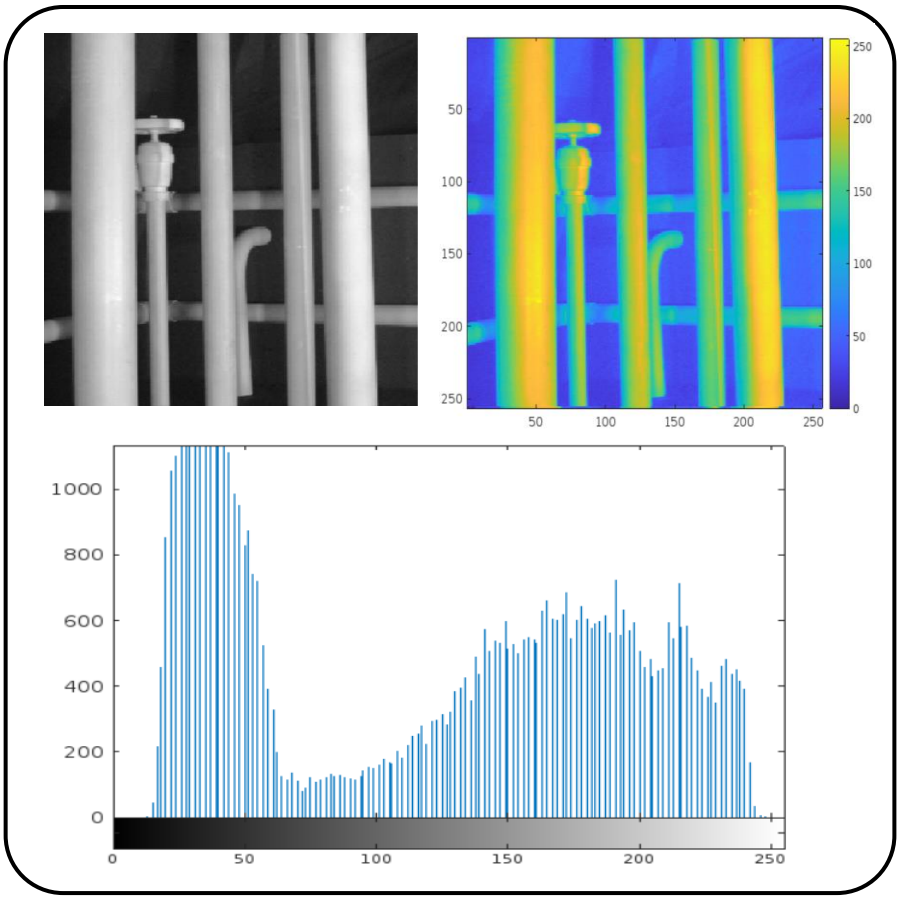


*Image **normalisée** et son histogramme*

- Avant la Normalisation :

L'mage était plus sombre, et l'histogramme était limité à une petite plage de valeurs, ce qui signifie que la contraste est faible.

- Après la Normalisation :

L'mage gagne en contraste, avec des noirs plus profonds et des blancs plus éclatants, et l'histogramme est étalé sur toute la plage [0, 255], ce qui indique une utilisation complète de la dynamique de l'image.

Cette transformation (*Normalisation*), améliore la visibilité des détails et le contraste global en étirant les valeurs de pixels sur toute la plage de l'intensité.

Comparaison avant / après L'égalisation :

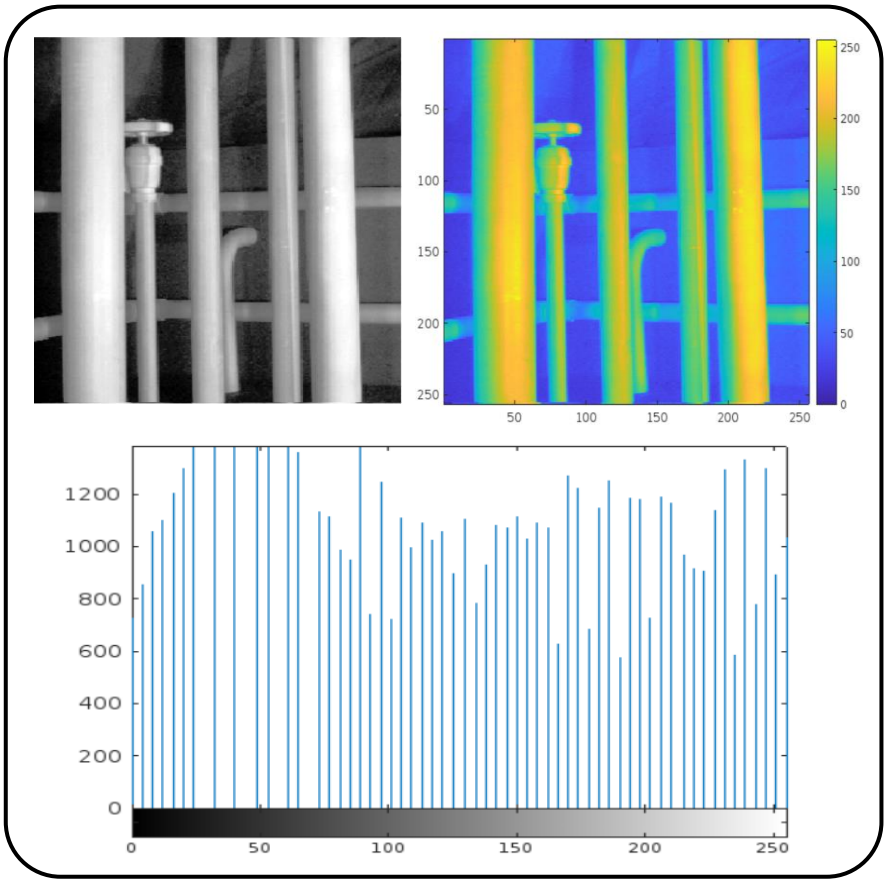


*Image **égalisée** et son histogramme*

- Avant l'égalisation :

L'mage dans sa forme originale a un faible contraste et elle est sombre avec des zones gris moyen, et son histogramme est concentré en un seul endroit indique aussi un faible contraste

- Après l'égalisation :

L'mage affichera un contraste amélioré, où les détails des zones sombres et claires sont plus visibles. Et concernant son histogramme, il est plus étalé, couvrant une plus grande plage de niveau de gris, ce qui indique que les intensités ont été redistribuées pour utiliser toute la dynamique possible

Conclusions sur la Normalisation et l'Égalisation d'histogramme :

La normalisation et l'égalisation d'histogramme sont deux techniques de traitement d'images visant à améliorer le contraste, mais elles différent dans leurs méthodes et objectifs (comme nous avons vu dans le cours) :

- Normalisation : utiliser au mieux l'échelle des niveaux (appelée aussi *expansion de dynamique*)
- Egalisation : équilibrer au mieux les niveaux, idéalement on cherche à obtenir un histogramme plat (donc histogramme cumulé sera droite).

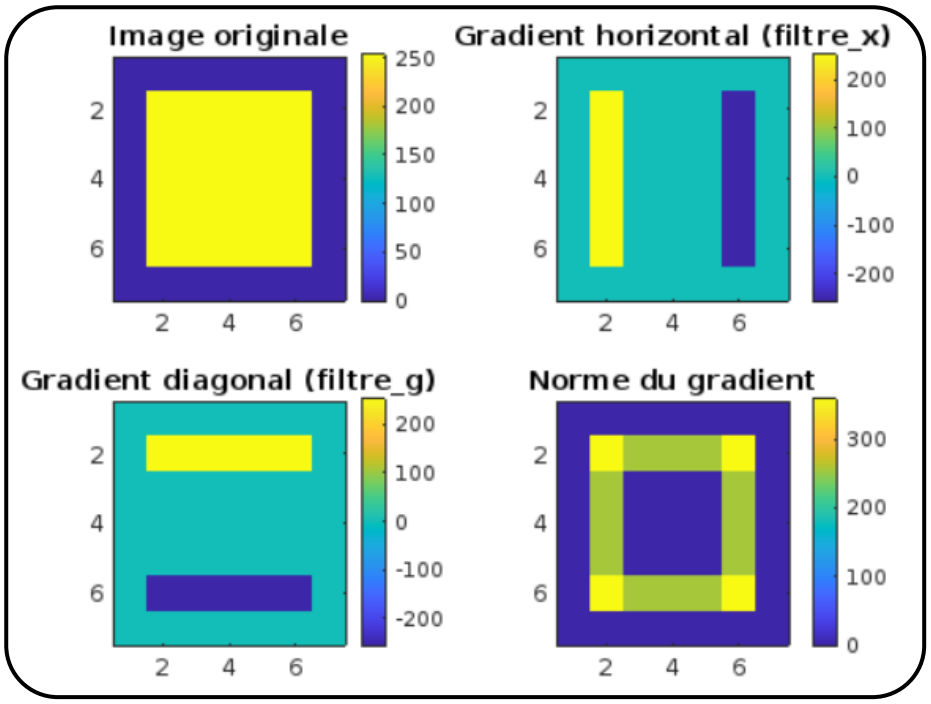
2 Filtrage

2.1 Un algorithme simple de détection de contours

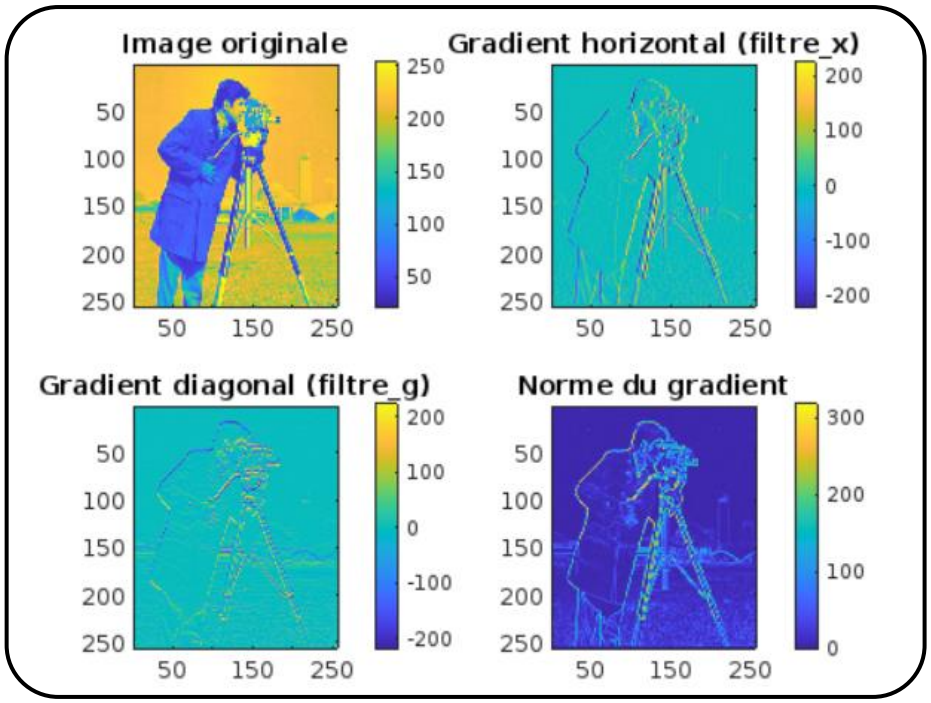
Commentaires :

Le gradient mesure le changement d'intensité dans l'image. Ici le filtre ∇_x capture les variations horizontale (contours verticaux) et le filtre ∇_y capture les variations diagonales.

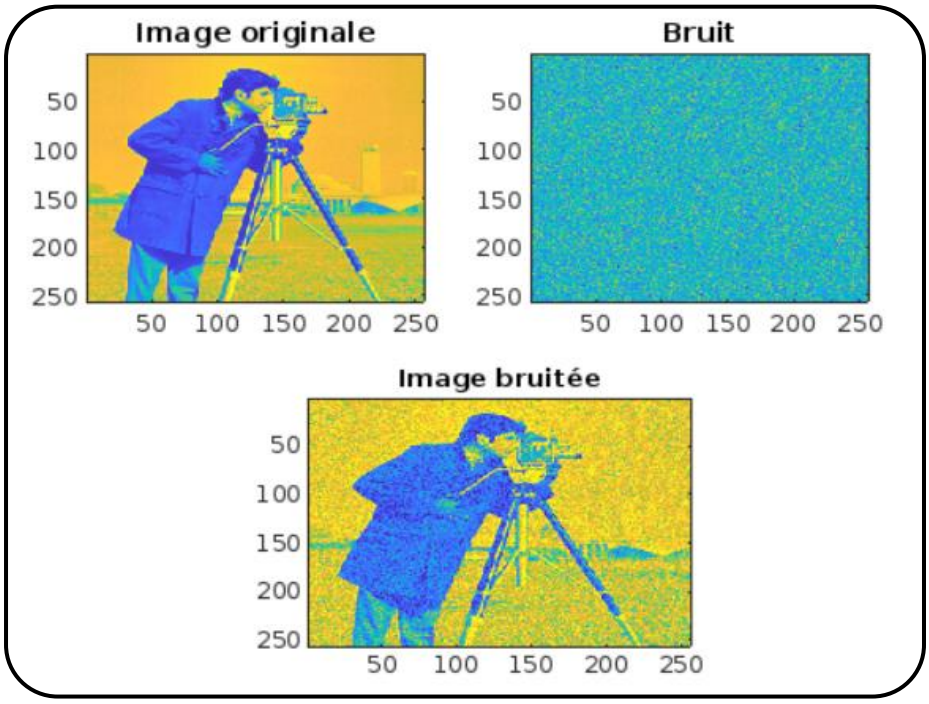
On remarque qu'avant le calcul de la norme du gradient, les résultats de convolution avec les filtres ∇_x et ∇_y contiennent des valeurs qui peuvent être négatives ou positives (allant de -225 à 255). Cependant, après le calcul de la norme du gradient, toutes les valeurs deviennent positives. Cela nous donne une représentation claire de la force des contours.



Et pour l'image '*cameraman.pgm*' :

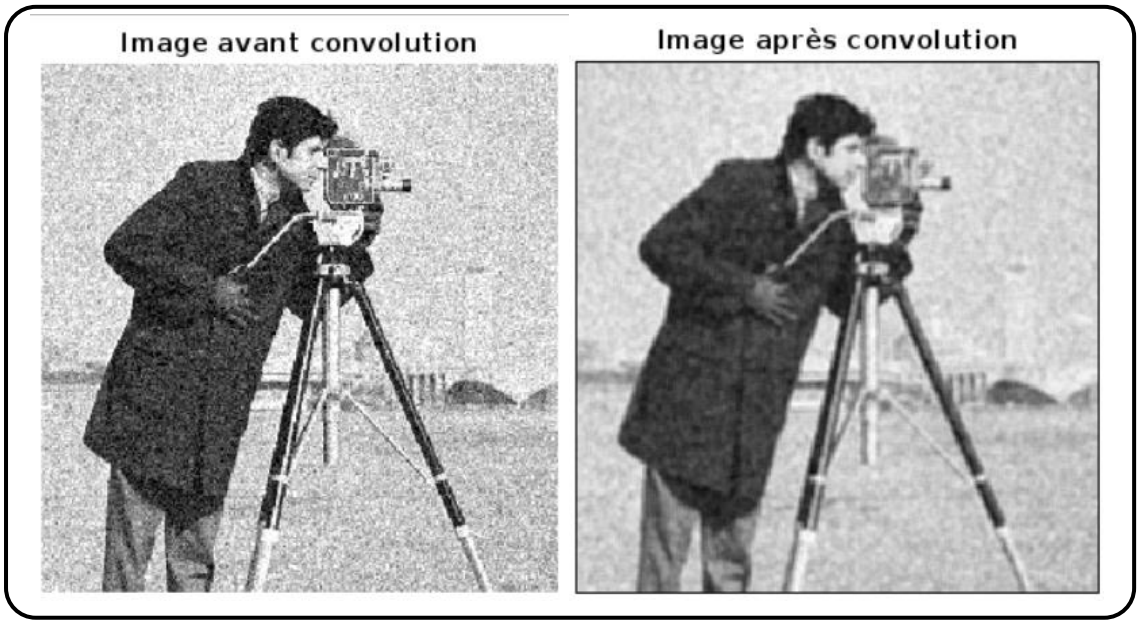


2.2 Robustesse au bruit



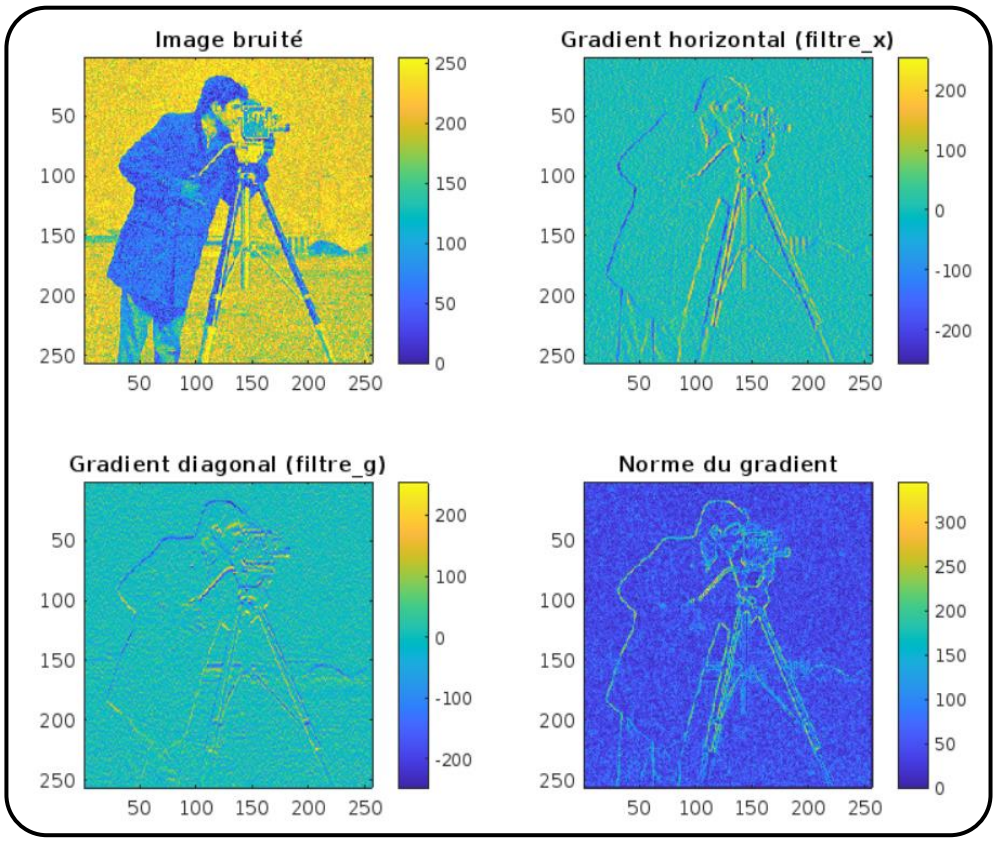
Pour l'image bruitée :

Implémentation de l'opération de convolution d'une image avec un filtre 3 ∗ 3 sans gérer les problèmes de bord :



Lorsqu'on applique un filtre de taille 3×3 (Ici nous avons choisi un filtre moyenneur) sur une image, on effectue une moyenne locale des valeurs de pixels dans un voisinage de 3×3 autour de chaque pixel. Cela a pour effet de réduire les variations locales, ce qui produit un lissage de l'image (un filtre moyenneur est efficace pour *lisser l'image* et réduire le bruit, mais il peut réduire la netteté des contours et des détails fins comme on a vu dans notre figure).

Convolution l'image bruitée avec les filtres gradient ∇_x et ∇_y *:*



On remarque que le filtre ∇x met en évidence les changements d'intensité horizontaux dans l'image. On peut voir les contours verticaux (comme le bord du corps de la personne, le trépied, et d'autres détails). Et par contre pour le filtre ∇y (les changements d'intensité diagonales dans l'image c'est-à-dire les transitions d'intensité de haut en bas).

La norme du gradient combine les informations des gradients horizontaux et verticaux pour obtenir la totalité des contours. Cela donne une image où les contours sont plus clairs et détaillés, mais les effets du bruit sont toujours présents

Image originale 'cameraman.pgm' et l'image bruitée avec les filtres de Sobel Sx et Sy :

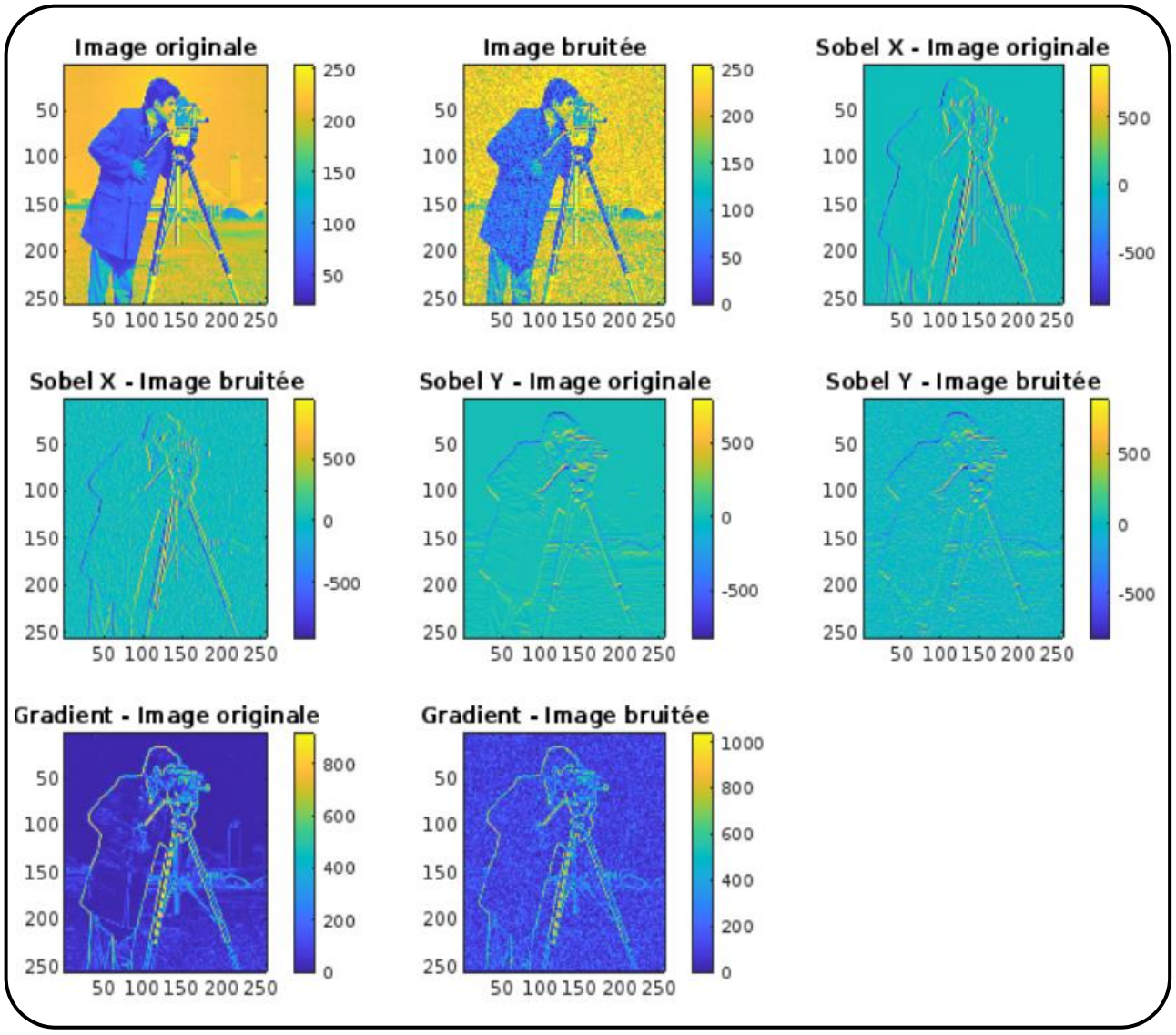


Image non bruitée (originale) :

Les filtres de *Sobel* détectent efficacement les contours dans l'image en calculant les gradients horizontal (Sx) et vertical (Sy) des intensités de pixels. En combinant les gradients Sx et Sy, on obtient la norme du gradient, qui montre les contours avec une plus grande clarté.

Image bruitée :

Dans l'image bruitée, les filtres de Sobel détectent non seulement les vrais contours, mais aussi les variations dues au bruit (Contours réels + bruit). Cela provoque des faux contours ou des points lumineux dans l'image, particulièrement dans les zones où le bruit est intense. C'est pour ça la norme du gradient dans l'image bruitée montre des contours réels, mais aussi de nombreuses perturbations. Ces perturbations rendent l'image plus difficile à interpréter visuellement,

Remarque :

L'application des filtres de Sobel sur une image bruitée fait apparaître les contours réels mais accentue également le bruit. Pour obtenir une détection de contours plus propre, il est souvent recommandé de lisser l'image (par exemple avec un filtre gaussien) pour réduire le bruit avant d'appliquer les filtres de Sobel.

Différences remarques entre les images de la norme du gradient et de la norme de Sobel :

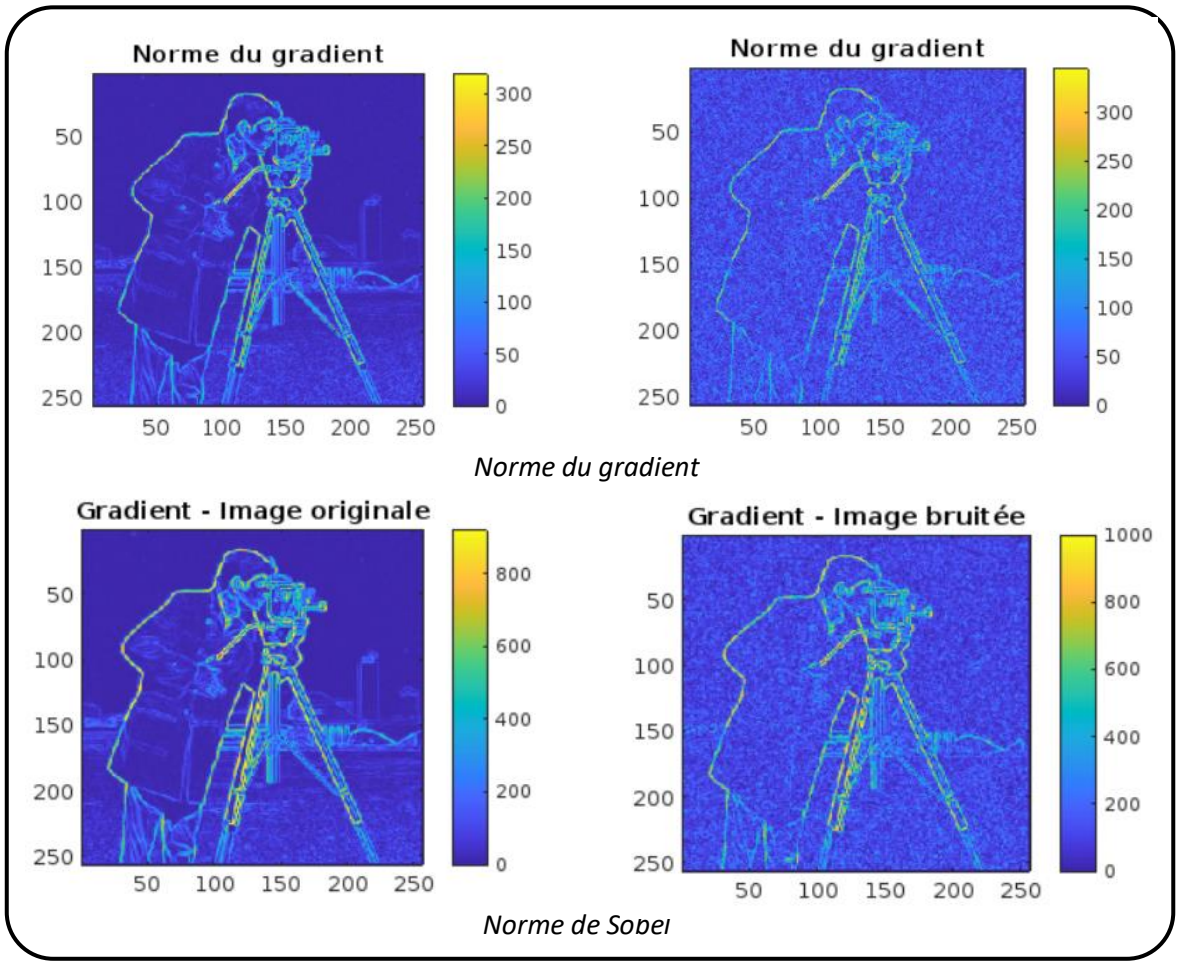


Image non bruitée (originale) :

- *Norme du gradient :* Les contours peuvent être visibles, mais l'image peut manquer de contraste dans les zones où l'intensité change moins brutalement (ne distingue pas bien entre les contours plus doux et les contours plus forts).
- *Norme de Sobel :* Le filtre de Sobel donne des contours plus nets et plus visibles, même dans les zones où l'intensité change progressivement. L'image obtenue avec la norme de Sobel présente des contours plus marqués et un contraste plus élevé que l'image avec la norme du gradient simple.

Image bruitée :

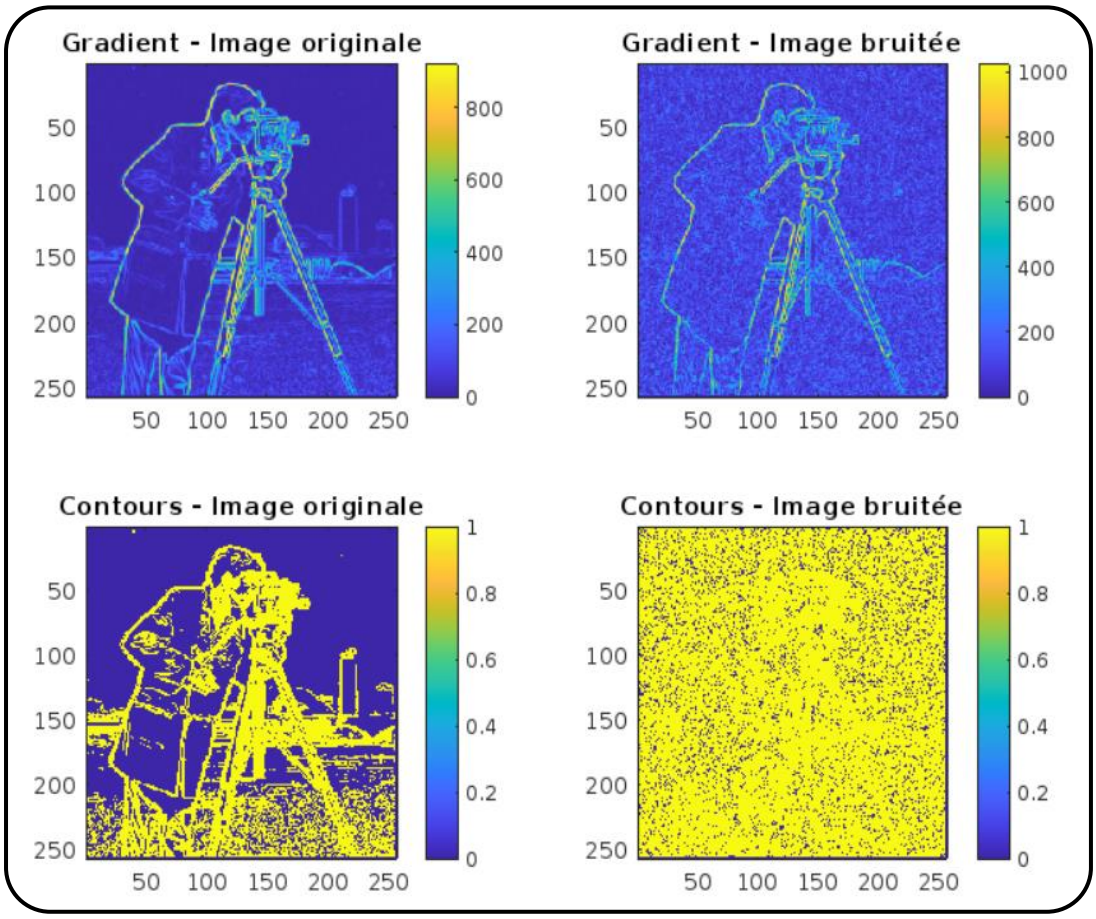
- *Norme du gradient :* Dans une image bruitée, la norme du gradient simple est très sensible au bruit, car chaque variation aléatoire d'intensité est prise en compte sans pondération. Cela se traduit par beaucoup de points aléatoires et de bruit dans l'image, ce qui rend difficile la distinction entre les contours réels et les effets du bruit.

- *Norme de Sobel :* Avec la norme de Sobel, Les variations faibles dues au bruit sont atténuées, ce qui permet d'obtenir des contours plus distincts et moins de bruit dans les zones sans contours. Bien que le bruit ne soit pas complètement éliminé, les contours sont mieux définis que dans l'image obtenue avec la norme du gradient simple, ce qui améliore la lisibilité de l'image.

Conclusion sur la robustesse au bruit des filtres gradients :

La robustesse au bruit des filtres gradient est un aspect crucial dans l'analyse d'images, notamment pour les applications de détection de contours.

2.3 Détection de contours



Remarque :

Dans l'image non bruitée, le seuillage de la norme du gradient est efficace pour extraire les contours. Par contre dans l'image bruitée, le seuillage de la norme du gradient montre ses limites, car il ne permet pas de filtrer entièrement le bruit sans compromettre la qualité des contours. Un prétraitement de l'image (par un filtrage passe-bas ou gaussien) est souvent nécessaire pour réduire le bruit avant d'appliquer le seuillage, afin d'obtenir des contours plus précis et moins perturbés par le bruit.

Annexes

1 Utilisation d'histogrammes

Affichage de l'image aquitaine.png ainsi que son histogramme :

```
image = imread('aquitaine.png');
figure;
subplot(4,2,1);
imshow(image); title('Image originale');
subplot(4,2,2);
imhist(image);
title("Histogramme de l'image originale");
subplot(4,2,[3.5,3.5]);
imagesc(image,[0 255]);colorbar();
title('Image originale')
image_modifiee = image + 100;
image_modifiee(image_modifiee>255) = 255; %ou min(image_modifiee = min(image+100,255));
subplot(4,2,5);
imshow(image_modifiee)
title('Image modifiée')
subplot(4,2,6);
imhist(image_modifiee)
title("Histogramme de l'image modifiée");
subplot(4,2,[7.5,7.5]);
imagesc(image_modifiee,[0 255]);;colorbar();
title('Image modifiée')
```

L'image en additionnant 100 et son histogramme :

```
image_modifiee = image + 100;
image_modifiee(image_modifiee>255) = 255;
%ou min(image_modifiee =min(image+100,255));
subplot(3,2,5);
imshow(image_modifiee)
title('Image modifiée')
subplot(3,2,6);
imhist(image_modifiee);
title("Histogramme de l'image modifiée");
```

Affichage de l'image lena.png et son histogramme (en divisant par 8 l'intensité de chaque pixel) :

```
image_2 = imread('lena.png');
subplot(4,2,1);
imshow(image_2)
title('Image originale');
subplot(4,2,2);
imagesc(image_2,[0 255]);colorbar();
title('Image originale')
subplot(4,2,3.5);
imhist(image_2)
title("Histogramme de l'image originale");
image_modifiee_2 = image_2 ./ 8;
subplot(4,2,5);
```

```
imshow(image_modifiee_2)
title('Image modifiée')
subplot(4,2,6);
imagesc(image_modifiee_2,[0 255]);colorbar();
title('Image originale')
subplot(4,2,7.5);
imhist(image_modifiee_2);
title("Histogramme de l'image modifiée");
```

Normalisation et Egalisation de l'histogramme :

```
image_3 = imread('micro1.png');
subplot(3,2,1)
imagesc(image_3, [0,255]);colorbar();
title('Image originale');
subplot(3,2,2)
imhist(image_3)
title("Histogramme de l'image originale");
%Normalisation de l'image
img_min = double(min(image_3(:)));
img_max = double(max(image_3(:)));
image_normalisee = uint8(255*(double(image_3)-img_min)/(img_max-img_min));
subplot(3,2,3)
%imshow(image_normalisee)
imagesc(image_normalisee, [0,255]);colorbar();
%title('Image normalisée');
subplot(3,2,4)
imhist(image_normalisee)
title("Histogramme de l'image normalisée");
%Egalisation de l'image
image_egalise = histeq(image_3);
subplot(3,2,5)
%imshow(image_egalise)
imagesc(image_normalisee, [0,255]);colorbar();
title('Image Egalisée');
subplot(3,2,6)
imhist(image_egalise)
title("Histogramme de l'image égalisée");
```

2 Filtrage

Création d'une image de carré blanc sur fond noir :

```
largeur = 10;
hauteur = 10;
image_noire = zeros(hauteur,largeur);
image_noire(3:8,3:8)=255;
subplot(2,2,1)
imshow(image_noire)
title('Carré blanc sur fond noir');
```

Convolution de l'image de carré avec chacun les filtres gradient :

```
% Fonction de convolution
function resultImage = applyConvolution(image, filter)
  [hauteur, largeur] = size(image);
  resultImage = zeros(hauteur, largeur);
    for i = 2:hauteur-1
    for j = 2:largeur-1
      region = image(i-1:i+1, j-1:j+1); % Extraire la région 3x3 autour du pixel
      resultImage(i, j) = sum(sum(region .* filter)); % Appliquer le filtre
    end
  end
end
% Programme principal
% Création d'une image carrée (par exemple, un carré blanc au centre sur fond noir)
image = zeros(7, 7);
image(2:6, 2:6) = 255;
% Définition des filtres de gradient
filtre_x = [0 0 0; -1 0 1; 0 0 0];  % Filtre pour le gradient horizontal
filtre_g = [0 -1 0; 0 0 0; 0 1 0];  % Filtre pour le gradient diagonal
% Convolution avec les filtres
image_x = applyConvolution(image, filtre_x);
image_g = applyConvolution(image, filtre_g);
% Calcul de la norme du gradient (combinaison des deux convolutions)
norme_gradient = sqrt(image_x.^2 + image_g.^2);
% Affichage des résultats
subplot(2,2,1), imagesc(image), title('Image originale');colorbar();
subplot(2,2,2), imagesc(image_x), title('Gradient horizontal (filtre\_x)');colorbar();
subplot(2,2,3), imagesc(image_g), title('Gradient diagonal (filtre\_g)');colorbar();
subplot(2,2,4), imagesc(norme_gradient), title('Norme du gradient');colorbar();
```

Pour l'image 'cameraman.pgm' :

```
% Fonction de convolution
function resultImage = applyConvolution(image, filter)
  [hauteur, largeur] = size(image);
  resultImage = zeros(hauteur, largeur);
    for i = 2:hauteur-1
    for j = 2:largeur-1
      region = image(i-1:i+1, j-1:j+1); % Extraire la région 3x3 autour du pixel
      resultImage(i, j) = sum(sum(region .* filter)); % Appliquer le filtre
    end
  end
end
% Programme principal
% Charger l'image cameraman
image = imread('cameraman.pgm');
image = double(image);  % Conversion en double pour permettre les calculs
% Définition des filtres de gradient
filtre_x = [0 0 0; -1 0 1; 0 0 0];  % Filtre pour le gradient horizontal
filtre_g = [0 -1 0; 0 0 0; 0 1 0];  % Filtre pour le gradient diagonal
% Convolution avec les filtres
```

```
image_x = applyConvolution(image, filtre_x);
image_g = applyConvolution(image, filtre_g);
% Calcul de la norme du gradient (combinaison des deux convolutions)
norme_gradient = sqrt(image_x.^2 + image_g.^2);
% Afficher les valeurs négatives
%image_x = abs(image_x);
%image_g = abs(image_g);
% Affichage des résultats
subplot(2,2,1), imagesc(image), title('Image originale');colorbar()
subplot(2,2,2), imagesc(image_x), title('Gradient horizontal (filtre\_x)');colorbar()
subplot(2,2,3), imagesc(image_g), title('Gradient diagonal (filtre\_g)');colorbar()
subplot(2,2,4), imagesc(norme_gradient), title('Norme du gradient');colorbar()
```

L'image 'cameraman.pgm' avec un bruit blanc :

```
% Charger l'image originale
image = imread('cameraman.pgm');
image = double(image);
% Ajouter du bruit blanc gaussien
sigma = 25; % Définir la variance du bruit
bruit = sigma * randn(size(image));
image_bruitee = image + bruit;
image_bruitee(image_bruitee > 255) = 255;
image_bruitee(image_bruitee < 0) = 0;
%Afficher l'image originale et l'image bruitée
figure;
subplot(2, 2, 1);
imagesc(image);
title('Image originale');
subplot(2, 2, 2);
imagesc(bruit);
title('Bruit');
subplot(2, 2, [3.5 3.6]);
imagesc(image_bruitee) ; title('Image bruitée');
```

Image bruitée et non bruitée avec le filtre de Sobel :

```
% Fonction de convolution
function resultImage = applyConvolution(image, filter)
  [hauteur, largeur] = size(image);
  resultImage = zeros(hauteur, largeur);
  for i = 2:hauteur-1
    for j = 2:largeur-1
      region = image(i-1:i+1, j-1:j+1); % Extraire la région 3x3 autour du pixel
      resultImage(i, j) = sum(sum(region .* filter)); % Appliquer le filtre
    end
  end
end
image = imread('cameraman.pgm');
image = double(image);
sigma = 25;
```

```
bruit = sigma * randn(size(image));
image_bruitee = image + bruit;
image_bruitee(image_bruitee > 255) = 255;
image_bruitee(image_bruitee < 0) = 0;
filtre_Sx = [-1 0 1; -2 0 2; -1 0 1];  % Filtre de Sobel en x
filtre_Sy = [-1 -2 -1; 0 0 0; 1 2 1];  % Filtre de Sobel en y
image_x = applyConvolution(image, filtre_Sx);
image_g = applyConvolution(image, filtre_Sy);
image_bruitee_x = applyConvolution(image_bruitee, filtre_Sx);
image_bruitee_g = applyConvolution(image_bruitee, filtre_Sy);
% gradient pour les images d'origine et bruitée
gradient_image = sqrt(image_x.^2 + image_g.^2);
gradient_image_bruite = sqrt(image_bruitee_x.^2 + image_bruitee_g.^2);
subplot(3, 3, 1);
imagesc(image);
title('Image originale');colorbar();
subplot(3, 3, 2);
imagesc(image_bruitee);
title('Image bruitée');colorbar();
subplot(3, 3, 3);
imagesc(image_x);
title('Sobel X - Image originale');colorbar();
subplot(3, 3, 4);
imagesc(image_bruitee_x);
title('Sobel X - Image bruitée');colorbar();
subplot(3, 3, 5);
imagesc(image_g);
title('Sobel Y - Image originale');colorbar();
subplot(3, 3, 6);
imagesc(image_bruitee_g);
title('Sobel Y - Image bruitée');colorbar();
subplot(3, 3, 7);
imagesc(gradient_image);
title('Gradient - Image originale');colorbar();
subplot(3, 3, 8);
imagesc(gradient_image_bruite);
title('Gradient - Image bruitée');colorbar();
```

Détection de contours :

```
% Fonction de convolution
function resultImage = applyConvolution(image, filter)
  [hauteur, largeur] = size(image);
  resultImage = zeros(hauteur, largeur);
    for i = 2:hauteur-1
    for j = 2:largeur-1
      region = image(i-1:i+1, j-1:j+1); % Extraire la région 3x3 autour du pixel
      resultImage(i, j) = sum(sum(region .* filter)); % Appliquer le filtre
    end
  end
end
```

```
image = imread('cameraman.pgm');
image = double(image);
sigma = 25;
bruit = sigma * randn(size(image));
image_bruitee = image + bruit;
image_bruitee(image_bruitee > 255) = 255;
image_bruitee(image_bruitee < 0) = 0;
filtre_Sx = [-1 0 1; -2 0 2; -1 0 1];  % Filtre de Sobel en x
filtre_Sy = [-1 -2 -1; 0 0 0; 1 2 1];  % Filtre de Sobel en y
image_x = applyConvolution(image, filtre_Sx);
image_g = applyConvolution(image, filtre_Sy);
image_bruitee_x = applyConvolution(image_bruitee, filtre_Sx);
image_bruitee_g = applyConvolution(image_bruitee, filtre_Sy);
% gradient pour les images d'origine et bruitée
gradient_image = sqrt(image_x.^2 + image_g.^2);
gradient_image_bruite = sqrt(image_bruitee_x.^2 + image_bruitee_g.^2);
seuil = 50;
contours_image = gradient_image > seuil;
contours_image_bruite = gradient_image_bruite > seuil;
subplot(2,2,1);
imshow(uint8(gradient_image), []);
title('Gradient - Image originale');
subplot(2,2,2);
imshow(uint8(gradient_image_bruite), []);
title('Gradient - Image bruitée');
subplot(2,2,3);
imshow(uint8(contours_image), []);
title('Contours - Image originale');
subplot(2,2,4);
imshow(uint8(contours_image_bruite), []);
title('Contours - Image bruitée');
```