

Base du traitement d'images

TDs des jours 10_11 et 28_11_23

Professeur : Didier Wolf

étudiant : Déric Augusto FRANÇA DE SALES

email : francade1u@etu.univ-lorraine.fr

Ce document a été généré à partir d'un fichier Live Script dans la version R2020a du logiciel MATLAB.

sommaire

Introduction.....	1
Transformation RGB -> HSV.....	2
Démonstration des LUT d'affichage, fausse couleur.....	5
Manipulation de l'histogramme.....	9
Filtrage passe-bas.....	13
Autres techniques de filtrage.....	17
Filtrage par addition.....	18
Filtrage linéaire.....	20
Filtrage médian.....	21
Filtrage bilatéral.....	22
Détection de contours.....	24
Détections de formes.....	29
Détection des caractères	42

Introduction

Le traitement d'images numériques est une discipline fondamentale dans le domaine de la technologie moderne, jouant un rôle crucial dans diverses applications allant de l'analyse médicale à la reconnaissance faciale. La transformation de l'espace colorimétrique RGB en HSV, par exemple, est une technique clé qui facilite la manipulation et l'interprétation des composantes de couleur dans les images. Cette conversion rend les variations de couleur plus perceptibles et moins sensibles aux variations de luminosité, ce qui est particulièrement utile pour des tâches comme la segmentation d'images ([MathWorks](#), [Lindevs](#)).

Par ailleurs, le filtrage d'images, y compris les techniques de filtrage passe-bas, linéaire, médian et bilatéral, est essentiel pour améliorer la qualité des images en réduisant le bruit et en accentuant les détails. Ces méthodes sont largement utilisées dans des domaines tels que l'imagerie médicale et la reconnaissance faciale, où la clarté et la précision des images sont primordiales. Le filtrage médian, en particulier, est connu pour sa capacité à préserver les bords tout en réduisant le bruit, ce qui est crucial pour l'analyse précise des images médicales ([Scikit-Image](#)).

Enfin, la détection de contours, de formes et de caractères est une composante essentielle de la vision par ordinateur. Ces techniques permettent d'extraire des informations pertinentes à partir d'images, facilitant des applications telles que la reconnaissance automatique de plaques d'immatriculation et l'OCR pour la numérisation de documents. La fiabilité et l'efficacité de ces méthodes ont un impact direct sur la performance des systèmes basés sur la vision par ordinateur, soulignant leur importance dans notre monde numérique.

et connecté. Ces progrès continuent de façonner des domaines variés, reflétant l'importance cruciale de la recherche et du développement dans le traitement d'images.

Ce compte rendu se concentrera sur l'analyse des techniques clés de traitement d'images numériques, telles que la transformation RGB en HSV, les méthodes de filtrage d'images et la détection de caractéristiques, pour en souligner son applicabilité et utilisation sur MATLAB.

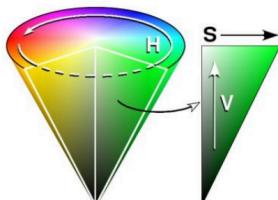
Transformation RGB -> HSV

La transformation de l'espace colorimétrique RGB (Rouge, Vert, Bleu) en HSV (Hue, Saturation, Value) est un processus qui réorganise la représentation des couleurs d'une image pour faciliter certaines analyses et manipulations. Dans le système RGB, les couleurs sont définies par leur mélange de rouge, vert et bleu. En revanche, le système HSV représente les couleurs en termes de teinte (Hue), saturation et valeur (brightness).

- **Teinte (Hue)** : Elle représente la couleur pure (sans ombre ni lumière) et est généralement exprimée en degrés autour d'un cercle de couleur. Par exemple, 0° correspond au rouge, 120° au vert, et 240° au bleu.
- **Saturation** : Elle indique l'intensité de la couleur. À saturation maximale (100%), la couleur est la plus pure, tandis qu'à saturation nulle (0%), la couleur serait complètement délavée, se rapprochant du gris.
- **Valeur (Value)** : Elle correspond à la luminosité de la couleur. Une valeur de 0% indique le noir, tandis que 100% représente la couleur la plus lumineuse possible.

Codage HSV

- Codage intuitif de la façon dont les couleurs sont perçues par l'œil.
- Teinte (Hue): désigne la couleur codée suivant son angle sur le cercle des couleurs
- Saturation: taux de pureté de la couleur, entre la couleur pure et le gris (achromatique)
- Value: brillance de la couleur, noir lorsque la valeur est nulle. Permet de jouer sur les contrastes.



Processus :

1. Trouver les valeurs maximales et minimales des composantes RGB

$$Max = \max(R, G, B)$$

$$Min = \min(R, G, B)$$

$$\Delta = Max - Min$$

2. Calcul de la Teinte (H)

$$H = \begin{cases} 0 & \text{si } \Delta = 0 \\ 60 \times \left(\frac{G-B}{\Delta} \bmod 6 \right) & \text{si } Max = R \\ 60 \times \left(\frac{B-R}{\Delta} + 2 \right) & \text{si } Max = G \\ 60 \times \left(\frac{R-G}{\Delta} + 4 \right) & \text{si } Max = B \end{cases}$$

3. Calcul de la Valeur (V)

$$V = Max$$

La transformation peut être faite sur MATLAB de la manière suivante :

```
clc; clear all; close all;

% Changer le dossier actuel pour l'endroit où se trouve le fichier MATLAB actuel
if(~isdeployed)
    cd(fileparts(matlab.desktop.editor.getActiveFilename));
end

img = imread('DAALE\dancing_battle.png');

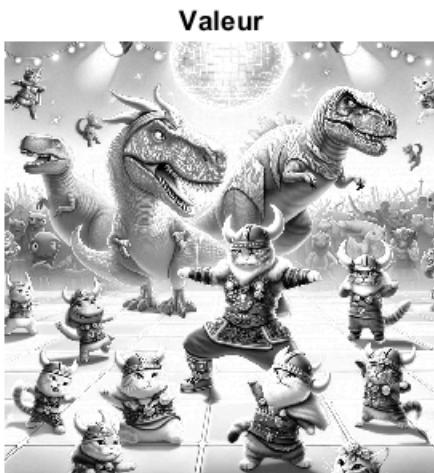
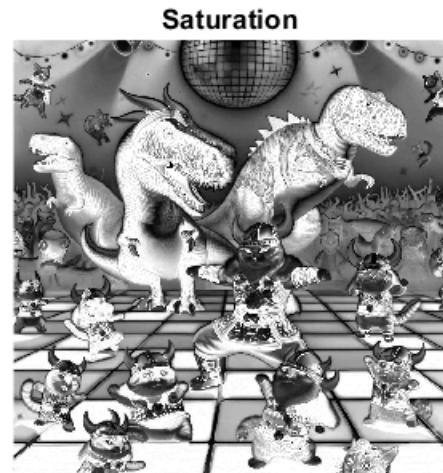
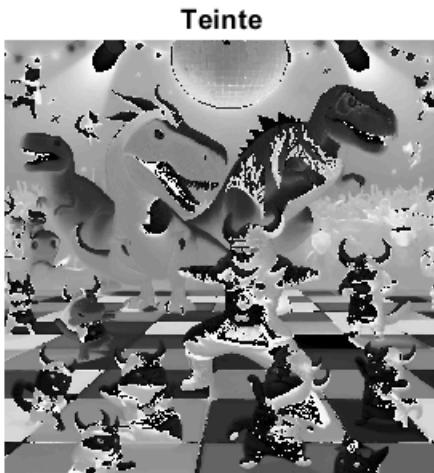
imshow(img), title('image generated from DAAL-E');
```

image generated from DAAL-E



```
img_hsv = rgb2HSV(img);

subplot(2,2,1), imshow(img_hsv(:,:,1)), title('Teinte');
subplot(2,2,2), imshow(img_hsv(:,:,2)), title('Saturation');
subplot(2,2,3), imshow(img_hsv(:,:,3)), title('Valeur');
subplot(2,2,4), imshow(img), title('Image Originale');
```



Démonstration des LUT d'affichage, fausse couleur

Le script en question illustre l'importance et l'utilité des tables de correspondance de couleurs (LUT) dans le traitement d'images numériques, en particulier dans la visualisation et l'analyse d'images. Les LUT sont essentielles pour transformer les données d'image brute en représentations visuelles plus significatives et informatives. En appliquant différentes colormaps comme 'gray', 'jet', 'winter', et 'hot' à une image en niveaux de gris, chaque LUT modifie la manière dont les valeurs de pixels sont mappées aux couleurs, révélant ainsi des détails et des caractéristiques qui ne sont pas immédiatement visibles dans l'image originale.

Ces transformations en fausse couleur sont cruciales dans des domaines tels que l'imagerie médicale, l'astronomie et la recherche géologique, où elles permettent de distinguer et d'analyser des caractéristiques subtiles. Par exemple, une LUT 'hot' peut être utilisée pour accentuer les régions de haute intensité, tandis que

'winter' peut mettre en évidence les contrastes à basse intensité. L'utilisation de ces différentes colormaps offre une flexibilité significative dans l'analyse des images, améliorant la perception des détails et aidant à extraire des informations précieuses qui seraient autrement difficiles à discerner.

```
clc; clear all; close all;

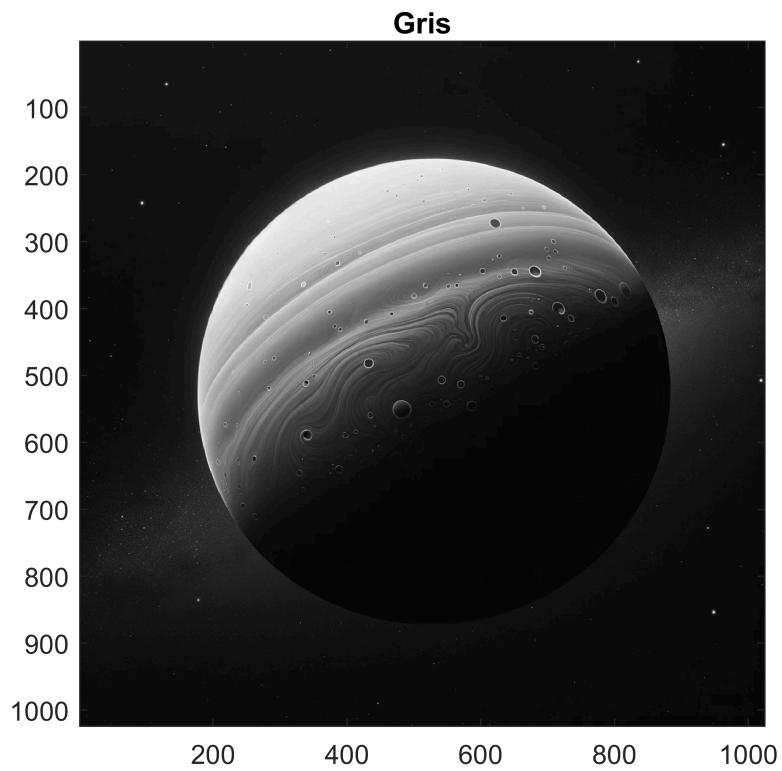
img2 = imread('DAALE\planet.png');
imshow(img2), title('image 2 generated from DAAL-E');
```

image 2 generated from DAAL-E

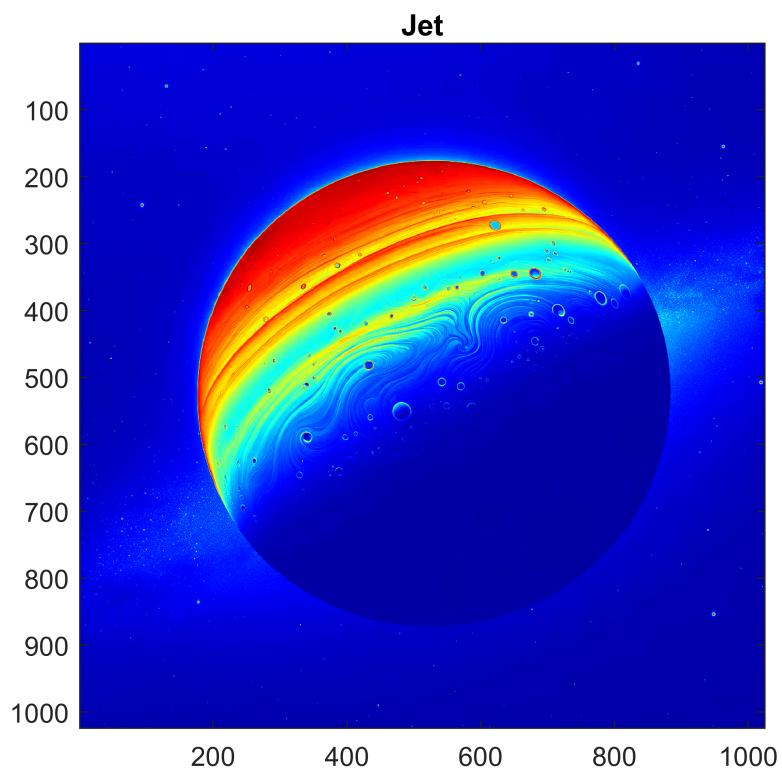


```
img2 = rgb2gray(img2);

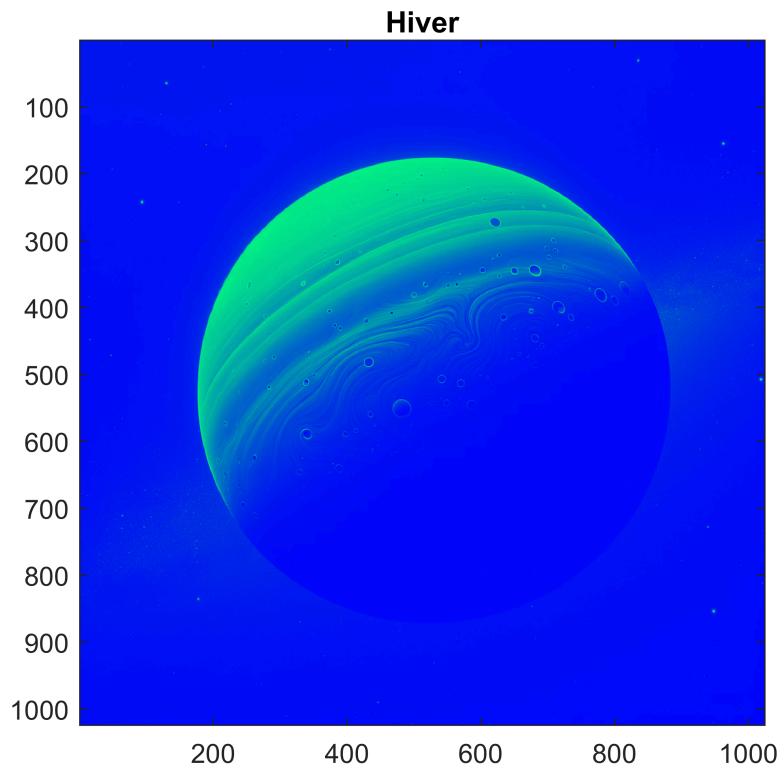
% Afficher avec la colormap 'gray'
figure, imagesc(img2), colormap('gray'), title('Gris'), axis image;
```



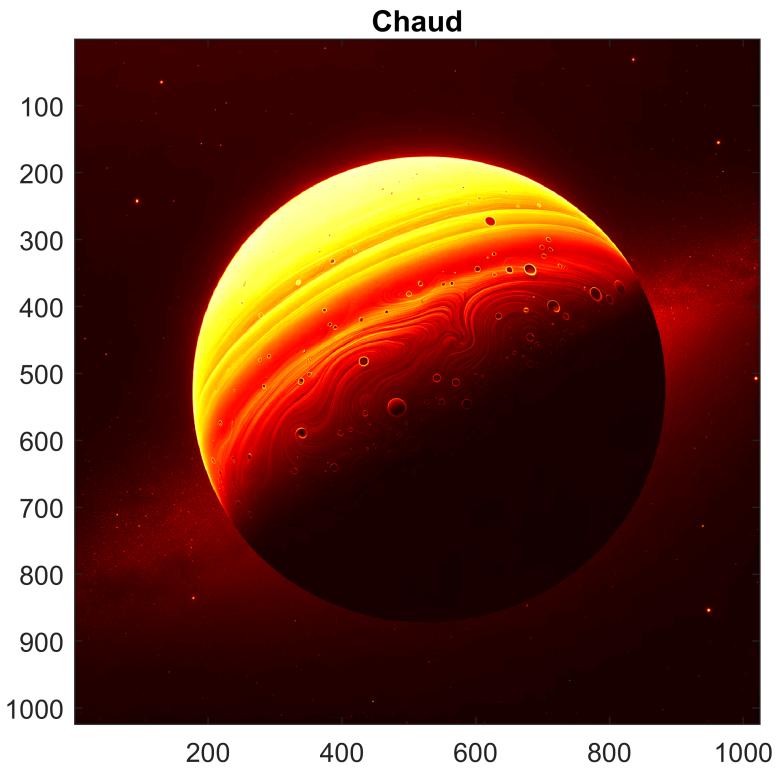
```
% Afficher avec la colormap 'jet'  
figure, imagesc(img2), colormap('jet'), title('Jet'), axis image;
```



```
% Afficher avec la colormap 'winter'  
figure, imagesc(img2), colormap('winter'), title('Hiver'), axis image;
```



```
% Afficher avec la colormap 'hot'  
figure, imagesc(img2), colormap('hot'), title('Chaud'), axis image;
```



Manipulation de l'histogramme

Dans le traitement d'images, la manipulation de l'histogramme est une méthode cruciale pour améliorer la qualité visuelle des images. L'égalisation d'histogramme, une des techniques clés, se concentre sur la redistribution uniforme des intensités lumineuses de l'image. Cette méthode est efficace pour augmenter le contraste global, surtout dans les images avec un contraste initial faible. Elle permet de révéler des détails dans des zones qui étaient auparavant trop sombres ou trop claires, ce qui est essentiel pour améliorer la clarté et la perception des détails.

Par ailleurs, l'égalisation d'histogramme adaptative représente une avancée significative dans cette approche. Contrairement à l'égalisation standard, elle ajuste le contraste de manière locale en traitant séparément de petites régions de l'image. Cette technique assure un ajustement plus précis et personnalisé du contraste, ce qui est particulièrement bénéfique dans les images présentant des variations de luminosité complexes. Ce processus permet de maintenir un aspect naturel tout en améliorant la clarté, rendant ces méthodes indispensables dans des applications variées, allant de l'art photographique à l'imagerie médicale.

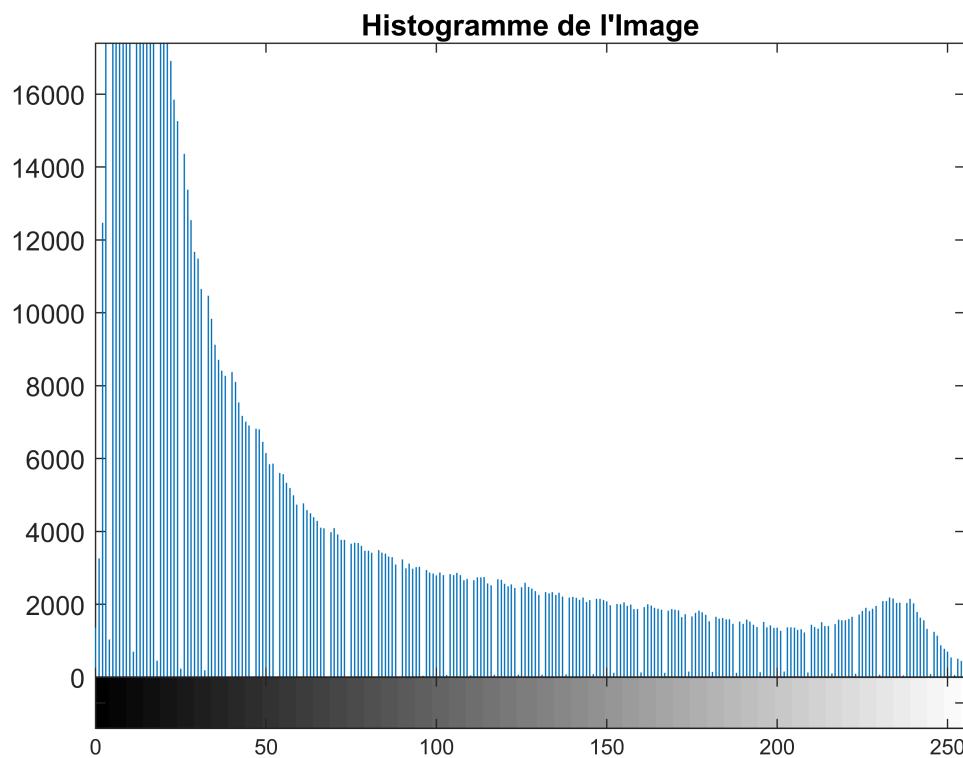
```
clear all;
```

```
img3 = rgb2gray(imread('DAALE\60s_child.png'));
figure; imshow(img3), title('image 3 générée par DAAL-E');
```

image 3 générée par DAAL-E



```
figure; imhist(img3); title('Histogramme de l''Image');
```



```

figure;
img3_eq = histeq(img3);
img3_adapt_eq = adapthisteq(img3);
img3_adjust = imadjust(img3);

colormap(gray);

subplot(2,2,1); imagesc(img3);
axis off; axis image; title('Image Originale');

subplot(2,2,2); imagesc(img3_eq);
axis off; axis image; title('Égalisation d''Histogramme');

subplot(2,2,3); imagesc(img3_adapt_eq);
axis off; axis image; title('Égalisation d''Histogramme Adaptative');

subplot(2,2,4); imagesc(img3_adjust);
axis off; axis image; title('Ajustement de l''Image');

set(gcf, 'Position', [100, 100, 1000, 800]);

```

Image Originale



Égalisation d'Histogramme



Égalisation d'Histogramme Adaptative



Ajustement de l'Image



Les résultats démontrés par les images et l'histogramme associé fournissent un aperçu clair de l'efficacité des méthodes de traitement d'image dans l'amélioration de la répartition des intensités lumineuses. L'histogramme pré-traitement indique une concentration élevée de valeurs d'intensité dans les basses gammes, traduisant un contraste limité au sein de l'image originale. L'application de l'égalisation d'histogramme entraîne un rehaussement notable du contraste, comme en témoigne la distribution élargie des intensités, révélant ainsi des détails qui étaient auparavant masqués dans les plages moyennes.

L'introduction de l'égalisation d'histogramme adaptative permet d'aller au-delà des limitations de l'égalisation globale, en procédant à des ajustements ciblés qui préservent l'équilibre des hautes lumières et des ombres. Cette méthode affine l'amélioration du contraste en ajustant localement les intensités, évitant ainsi la sur- et sous-exposition des régions critiques de l'image. L'ajustement de l'image, quant à lui, réalise une extension des valeurs d'intensité sur la plage complète disponible, ce qui se traduit par une augmentation générale de la clarté et la mise en exergue des détails dans les zones sous-exposées.

Ces techniques de manipulation d'histogramme sont primordiales pour la préparation d'images en vue d'applications analytiques variées, allant de l'interprétation visuelle par des experts à l'analyse automatisée par des systèmes informatiques. Elles constituent un aspect fondamental de l'amélioration des images pour des domaines d'application tels que la télédétection, l'imagerie médicale et la restauration photographique, où la fidélité des détails et le contraste sont essentiels.

Filtrage passe-bas

Le filtrage passe-bas est une technique fondamentale en traitement d'images, utilisée pour réduire le bruit et améliorer la qualité visuelle des images. Dans le contexte présenté, l'image originale en niveaux de gris est intentionnellement corrompue avec différents types de bruit (gaussien, sel et poivre, speckle) pour simuler des conditions de qualité dégradée. Chaque type de bruit introduit des perturbations caractéristiques, comme des variations aléatoires de l'intensité pour le bruit gaussien ou des pixels isolés anormalement clairs ou sombres pour le bruit sel et poivre.

Pour atténuer ces bruits, des filtres passe-bas sont appliqués, spécifiquement le filtre gaussien et le filtre moyen, tous deux avec une fenêtre de convolution de taille 3x3. Le filtre gaussien utilise une distribution gaussienne pour pondérer les pixels voisins, atténuant le bruit tout en préservant les bords et détails de l'image. Le filtre moyen, en revanche, attribue un poids égal à chaque voisin, ce qui peut entraîner une légère perte de netteté. Ces filtres sont appliqués sur les images bruitées pour produire des versions filtrées qui sont ensuite affichées pour comparaison. Les réponses fréquentielles de ces filtres, obtenues par l'outil `freqz2`, illustrent comment chaque filtre atténue les fréquences élevées associées au bruit, avec des différences notables entre le profil doux du filtre gaussien et celui plus uniforme du filtre moyen. Ces caractéristiques les rendent appropriés pour diverses applications de réduction de bruit en traitement d'images.

```
clc; clear all; close all;

% Charger l'image et la convertir en niveaux de gris
img4 = rgb2gray(imread('DAALE\cameraman.png'));
figure; imshow(img4), title('image 4 générée par DAAL-E');
```

image 4 générée par DAAL-E



```
% Définir les types de bruit
noises = {'gaussian', 'salt & pepper', 'speckle'};
noisy_images = cell(1, length(noises));
gaussian_filtered_images = cell(1, length(noises));
average_filtered_images = cell(1, length(noises));

% Ajouter du bruit à l'image
for i = 1:length(noises)
    switch noises{i}
        case 'gaussian'
            noisy_images{i} = imnoise(img4, 'gaussian', 0, 0.01);
        case 'salt & pepper'
            noisy_images{i} = imnoise(img4, 'salt & pepper', 0.01);
        case 'speckle'
```

```

        noisy_images{i} = imnoise(img4, 'speckle', 0.01);
    end
end

% Définir les filtres Gaussien et Moyen
gaussian_h = fspecial('gaussian', [3 3], 0.5);
average_h = fspecial('average', [3 3]);

% Appliquer les filtres Gaussien et Moyen aux images bruitées
for i = 1:length(noises)
    gaussian_filtered_images{i} = imfilter(noisy_images{i}, gaussian_h, 'replicate');
    average_filtered_images{i} = imfilter(noisy_images{i}, average_h, 'replicate');
end

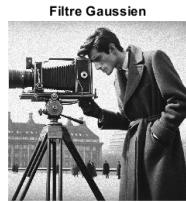
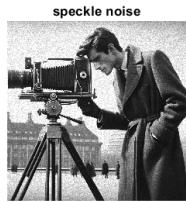
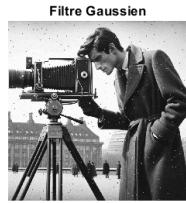
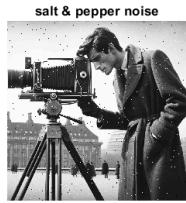
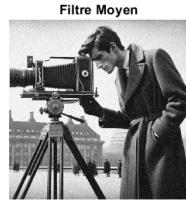
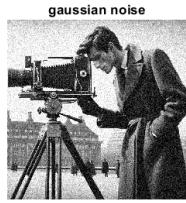
% Afficher les images bruitées et les images filtrées
figure;
for i = 1:length(noises)
    % Image bruitée
    subplot(length(noises), 4, (i-1)*4 + 1);
    imshow(noisy_images{i});
    title([noises{i} ' noise']);

    % Image filtrée par le filtre Gaussien
    subplot(length(noises), 4, (i-1)*4 + 2);
    imshow(gaussian_filtered_images{i});
    title('Filtre Gaussien');

    % Image filtrée par le filtre Moyen
    subplot(length(noises), 4, (i-1)*4 + 3);
    imshow(average_filtered_images{i});
    title('Filtre Moyen');
end

% Ajuster la disposition
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

```



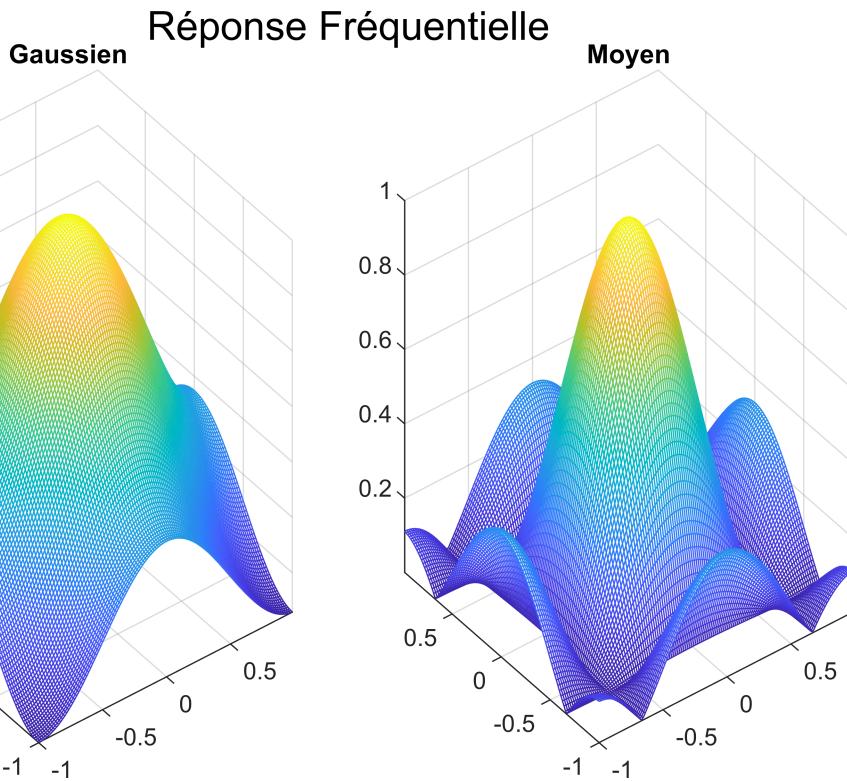
```
% Calculer les réponses fréquentielles en utilisant freqz2
[H_gaussian, f1_gaussian, f2_gaussian] = freqz2(gaussian_h, 128, 128);
[H_average, f1_average, f2_average] = freqz2(average_h, 128, 128);

% Afficher la réponse fréquentielle des filtres Gaussien et Moyen
figure; hold on;

% Agrandir les sous-graphiques en ajustant les axes
% Réponse fréquentielle du filtre Gaussien
subplot('Position', [0.05 0.1 0.4 0.8]); % Ajuster la position comme [gauche bas largeur hauteur]
mesh(f1_gaussian, f2_gaussian, abs(H_gaussian));
axis tight; % Ajuster l'axe étroitement aux données
title('Gaussien');

% Réponse fréquentielle du filtre Moyen
subplot('Position', [0.55 0.1 0.4 0.8]); % Ajuster la position comme [gauche bas largeur hauteur]
mesh(f1_average, f2_average, abs(H_average));
axis tight; % Ajuster l'axe étroitement aux données
title('Moyen');

% Ajouter un grand titre à la figure
sgtitle('Réponse Fréquentielle');
hold off;
```



Les images et les graphiques de réponse fréquentielle fournissent une visualisation concrète de l'efficacité des filtres passe-bas dans la réduction du bruit. Ces résultats démontrent la capacité de chaque filtre à restaurer la qualité de l'image tout en fournissant des indices sur leur sélectivité en fréquence et leur impact sur la préservation des détails.

Autres techniques de filtrage

En traitement d'images, diverses techniques de filtrage sont employées pour atténuer les effets du bruit. Le filtrage linéaire moyenneur, utilisant une fenêtre de 9x9 pixels, sert à réduire le bruit gaussien. Cette méthode lisse l'image en calculant la moyenne des pixels environnants, diminuant les variations induites par le bruit, tout en maintenant la structure générale. Pour le bruit de type sel et poivre, qui ajoute des pixels isolés de forte luminance ou obscurité, le filtrage médian est utilisé. Il remplace la valeur de chaque pixel par la médiane de ses voisins, ce qui efface efficacement ce type de bruit et conserve mieux les contours et textures originales de l'image. Ces méthodes de filtrage sont intégrées dans les processus de prétraitement des images pour l'amélioration de la qualité avant analyses subséquentes.

```
clc; clear all; close all;

% Charger l'image originale et la convertir en niveaux de gris
img5 = rgb2gray(imread('DAALE\bateau.png'));
figure; imshow(img5), title('image 5 générée par DAAL-E');
```

image 5 générée par DAAL-E



Filtrage par addition

```
% Créer une image bruitée par addition de bruit gaussien
figure; hold on;
colormap(gray);
img5double = im2double(img5);
img5bruit = imnoise(img5double, 'gaussian');
imshow(img5bruit);
title('Image Bruitée (Gaussien)');
axis off; % Retirer les axes
hold off;
```

Image Bruitée (Gaussien)



```
% Accumuler du bruit pour simuler une image très bruitée
figure; hold on;
for i = 1:30
    img5bruit = imnoise(img5double, 'gaussian') + img5bruit;
end

% Afficher l'image originale, l'image bruitée, et l'image très bruitée
subplot(1,3,1), imagesc(img5);
title('Original');
axis off; % Retirer les axes

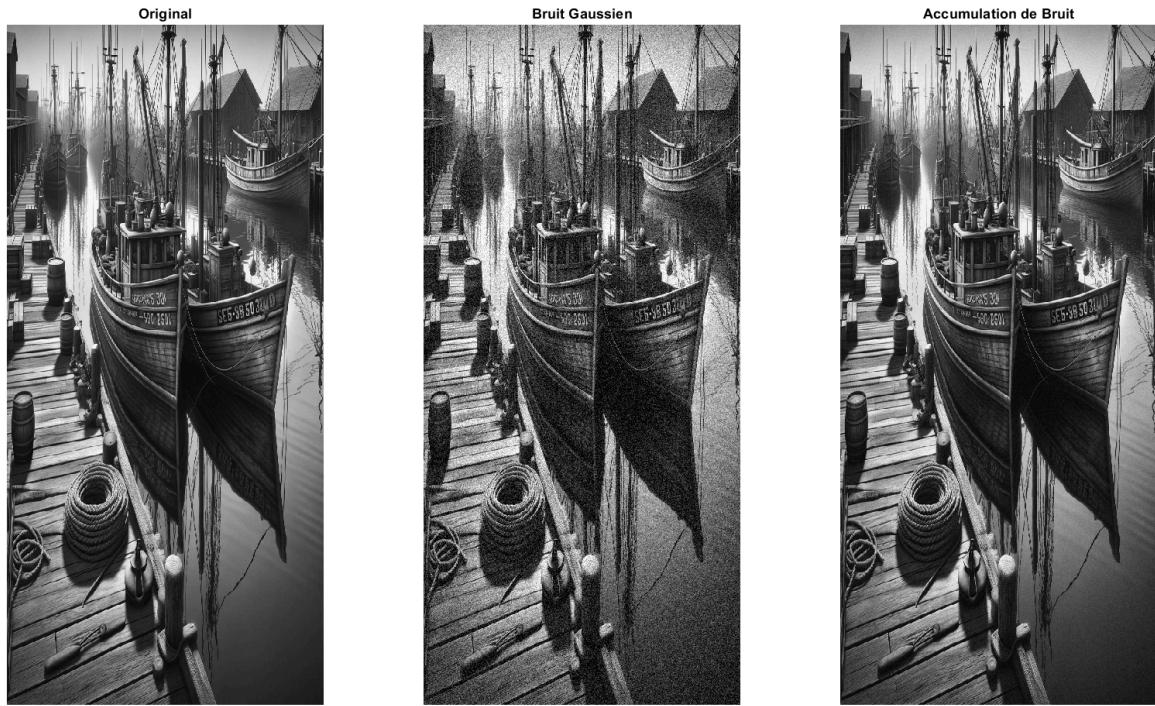
subplot(1,3,2), imagesc(imnoise(img5double, 'gaussian'));
title('Bruit Gaussien');
axis off; % Retirer les axes
```

```

subplot(1,3,3), imagesc(img5bruit);
title('Accumulation de Bruit');
axis off; % Retirer les axes

% Ajuster la disposition
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
colormap(gray);
hold off;

```



Filtrage linéaire

```

% Appliquer un filtre linéaire moyenneur
figure; hold on;
img5bruit = imnoise(img5double, 'gaussian');
filt = fspecial('average', [9,9]);
img5bruitfiltre = imfilter(img5bruit, filt);

% Afficher l'image bruitée et l'image filtrée
subplot(1,2,1), imagesc(img5bruit);
title('Bruit Gaussien');
axis off; % Retirer les axes

subplot(1,2,2), imagesc(img5bruitfiltre);
title('Filtrage Moyenneur');
axis off; % Retirer les axes

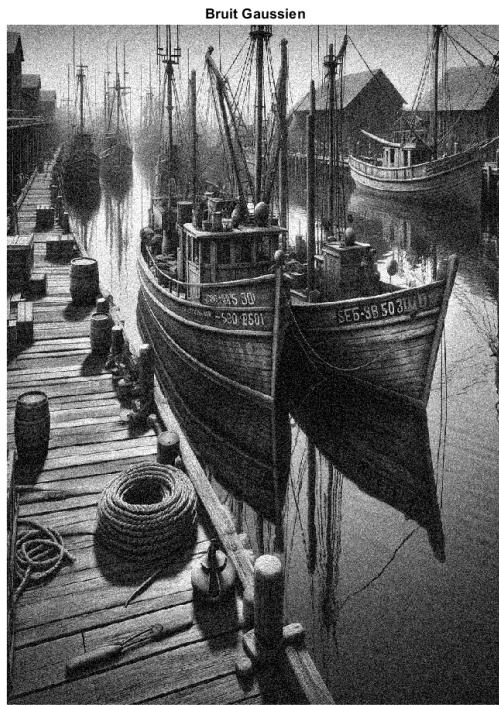
% Ajuster la disposition

```

```

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
colormap(gray);
hold off;

```



Filtrage médian

```

% Appliquer un filtrage médian sur une image bruitée "sel et poivre"
figure; hold on;
colormap(gray);
img5bruit = imnoise(img5, 'salt & pepper');
imagefiltree = medfilt2(img5bruit);

% Afficher l'image originale, l'image bruitée "sel et poivre", et l'image filtrée
subplot(1,3,1), imagesc(img5);
title('Original');
axis off; % Retirer les axes

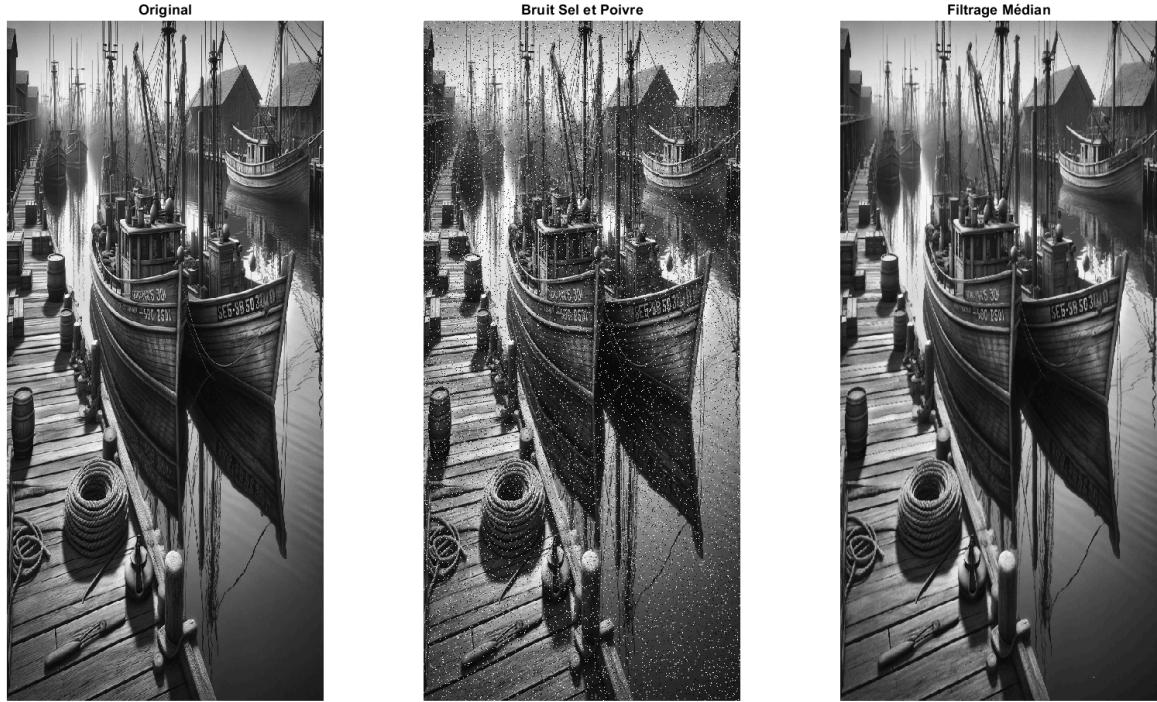
subplot(1,3,2), imagesc(img5bruit);
title('Bruit Sel et Poivre');
axis off; % Retirer les axes

subplot(1,3,3), imagesc(imagefiltree);
title('Filtrage Médian');
axis off; % Retirer les axes

% Ajuster la disposition
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

```

```
hold off;
```



Filtrage bilatéral

L'approche consiste à simuler une altération de la qualité d'image par l'introduction de bruit gaussien, suivi de l'application du filtrage bilatéral. Ce dernier se distingue par sa capacité à lisser le bruit tout en préservant les contours et les détails, opérant ainsi comme un compromis entre les méthodes de filtrage linéaire et non linéaire. Le filtrage bilatéral utilise la variance du bruit, obtenue à partir d'un patch de l'image bruitée, pour moduler l'intensité du lissage, permettant de conserver les formes et structures essentielles de l'image.

La répétition du processus de bruitage et de filtrage vise à affiner l'efficacité de la technique. L'évaluation par comparaison des images avant et après traitement met en évidence la capacité du filtrage bilatéral à éliminer le bruit sans sacrifier les détails significatifs. Cette méthode s'avère particulièrement utile dans les applications où la préservation des caractéristiques géométriques est indispensable, comme en imagerie médicale, reconnaissance de formes ou dans le cadre de la restauration d'images anciennes.

```
clc; clear all; close all;

% Charger l'image 'formes.bmp'
imgformes = imread('formes.bmp');

% Ajouter du bruit gaussien à l'image
imgformes_bruit = imnoise(imgformes, 'gaussian', 0, 0.04);
```

```
% Découper un patch de l'image bruitée pour déterminer le degré de bruit
patch = imcrop(imgformes_bruit, [1 1 40 40]);

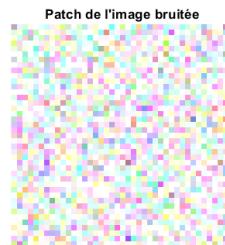
% Afficher le patch
figure; imshow(patch); hold on;
title('Patch de l''image bruitée');
%f1.WindowState = 'maximized';

% Calculer la variance du patch
patchVar = std2(patch)^2;

% Déterminer le degré de similarité pour le filtrage bilatéral
Dos = 5.*patchVar;

% Appliquer le filtrage bilatéral sur l'image bruitée
imagefiltree = imbilatfilt(imgformes_bruit, Dos, 4);
colormap(gray);

% Ajuster la disposition
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
hold off;
```



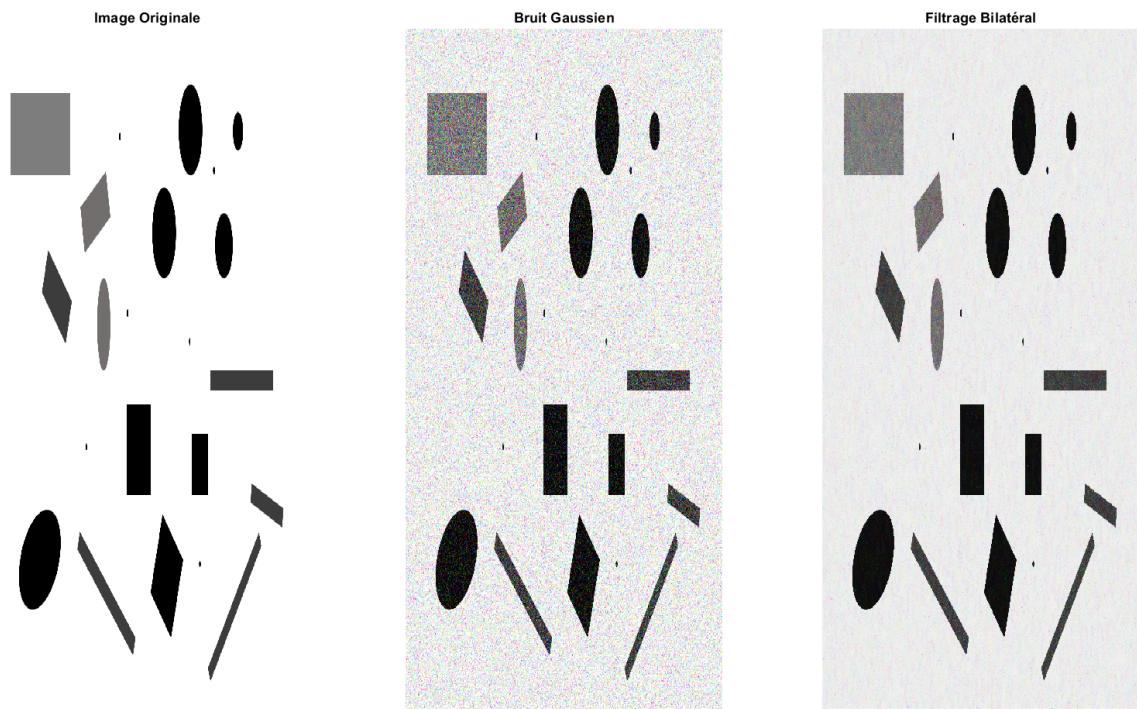
```
% Afficher l'image originale, l'image bruitée et l'image filtrée
f2 = figure();
f2.WindowState = 'maximized';
subplot(1,3,1), imagesc(imgformes); title('Image Originale'); axis off;
subplot(1,3,2), imagesc(imgformes_bruit); title('Bruit Gaussien'); axis off;
subplot(1,3,3), imagesc(imagefiltree); title('Filtrage Bilatéral'); axis off;
```

```

colormap(gray);

% Ajuster la disposition
set(gcf, 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

```



Détection de contours

L'approche algorithmique pour la détection et la classification de formes dans une image s'appuie sur la conversion de l'image en une représentation en niveaux de gris et son binarisation ultérieure. Cette binarisation est cruciale, car elle transforme l'image en une forme simplifiée qui met en contraste les régions d'intérêt avec leur arrière-plan, facilitant la détection des contours. L'algorithme d'Otsu joue un rôle clé dans ce processus, en déterminant de manière optimale le seuil qui sépare le plus efficacement les pixels correspondant aux formes de ceux de l'arrière-plan. La détection des contours est effectuée à l'aide de l'algorithme de Canny, une méthode éprouvée qui identifie les limites des formes avec précision et fiabilité.

Suite à l'identification des contours, l'analyse morphologique est réalisée pour caractériser et classer les formes selon leur géométrie. Les mesures d'aire, de périmètre et des axes dimensionnels fournissent des données quantitatives qui sont ensuite utilisées pour calculer l'ellipticité et la circularité. Ces mesures mathématiques permettent de distinguer entre des formes géométriques standard telles que les cercles, qui présentent une grande circularité et une ellipticité proche de l'unité, et des formes plus allongées comme les ellipses. Cette capacité de classification automatique des formes dans les images est indispensable dans le domaine de la

vision par ordinateur, où elle trouve des applications diverses telles que la navigation robotique, l'interprétation d'imagerie médicale et la surveillance par caméra.

```
clc; clear all; close all;

% Charger l'image
img = imread('formes.bmp');

% Convertir en niveaux de gris
img_gray = rgb2gray(img);

% Seuiller l'image pour créer une image binaire
threshold = graythresh(img_gray); % Utilisez l'algorithme d'Otsu pour trouver un seuil automatique
img_bw = imbinarize(img_gray, threshold);

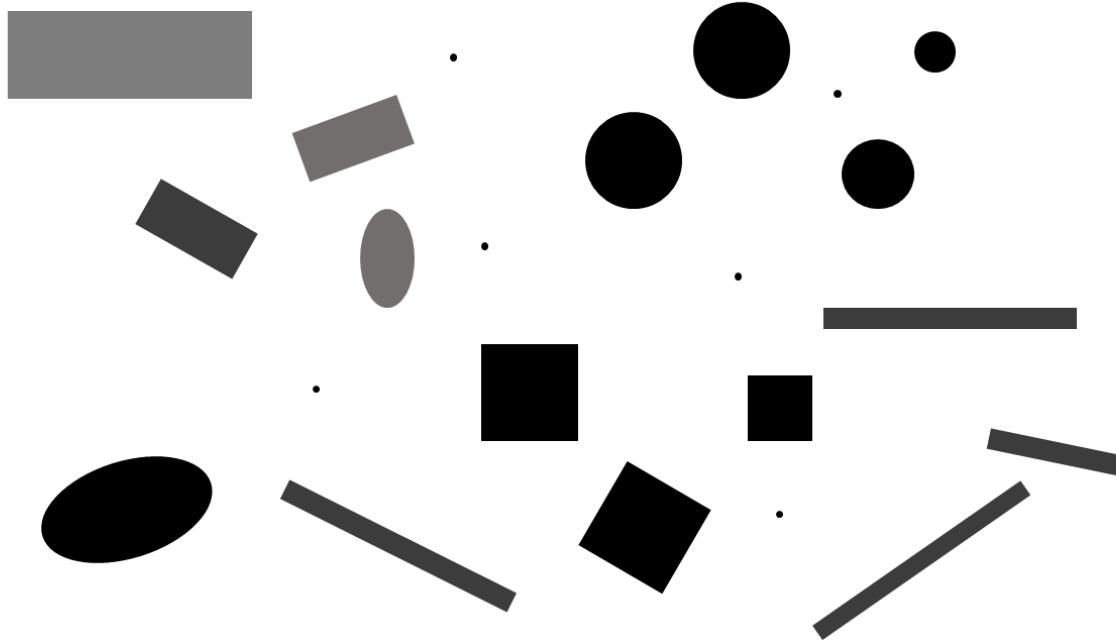
% Déetecter les contours dans l'image
contours = edge(img_bw, 'Canny');

% Trouver les contours fermés (formes)
[formes, ~] = bwboundaries(contours, 'noholes');

% Compter le nombre de formes
nombre_de_formes = length(formes);

% Afficher le résultat
imshow(img);
title(['Nombre de formes détectées : ', num2str(nombre_de_formes)]);
```

Nombre de formes détectées : 22

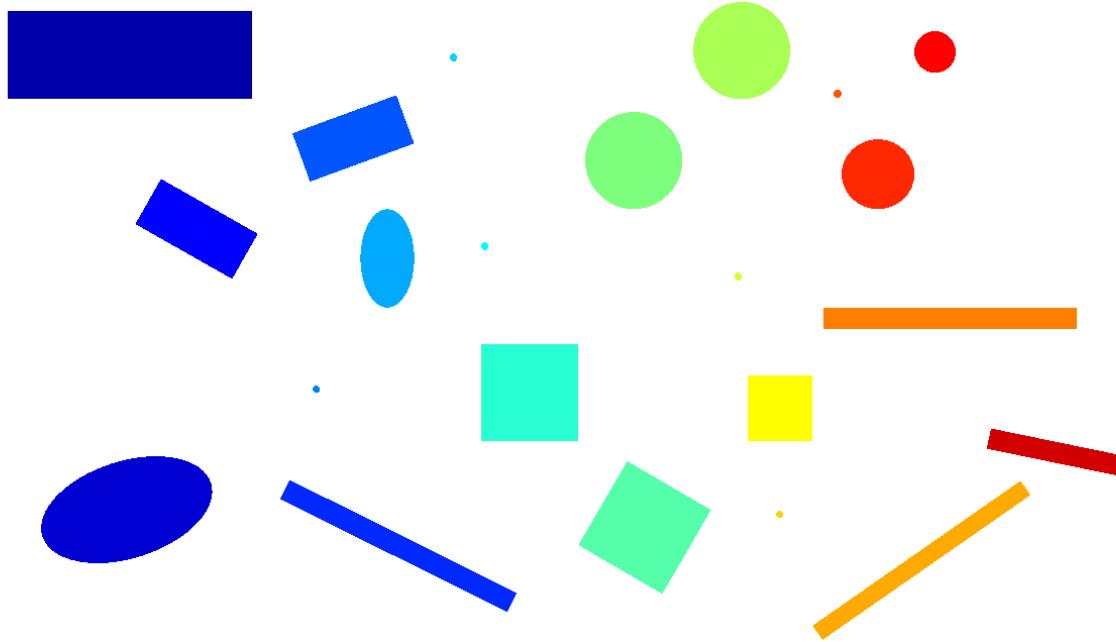


```
% Pour visualiser les formes détectées
figure; imshow(img_bw); hold on;
for k = 1:length(formes)
    boundary = formes{k};
    plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2);
end
```

```
% Inverser l'image binaire pour que les formes soient en blanc et le fond en noir
img_bw_inverted = ~img_bw;

% Utiliser bwlabel sur l'image inversée
[labeledImage, number_of_blobs] = bwlabel(img_bw_inverted, 4);

% Afficher le résultat
imshow(label2rgb(labeledImage));
title(['Nombre de formes détectées : ', num2str(number_of_blobs)]);
```



```
% Obtenir les propriétés des régions
blobMeasurements = regionprops(labeledImage, 'Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength');

% Analyser chaque région pour identifier la forme
resultats = strings(1, number_of_blobs);
for k = 1 : number_of_blobs
    % Calculer l'ellipticité (rapport entre les axes majeur et mineur)
    ellipticite = blobMeasurements(k).MajorAxisLength / blobMeasurements(k).MinorAxisLength;

    % Calculer la circularité
    circularite = (4 * pi * blobMeasurements(k).Area) / (blobMeasurements(k).Perimeter ^ 2);

    % Identifier la forme
    if circularite > 0.9 && ellipticite < 1.1
        resultats(k) = "Cercle";
    elseif ellipticite < 1.1
        resultats(k) = "Carré";
    elseif ellipticite >= 1.1 && ellipticite < 1.5
        resultats(k) = "Rectangle";
    else
        resultats(k) = "Ellipse";
    end
end

% Afficher les résultats
disp(resultats)
```

"Ellipse"

Column 2

"Ellipse"

Column 3

"Ellipse"

Column 4

"Ellipse"

Column 5

"Ellipse"

Column 6

"Cercle"

Column 7

"Ellipse"

Column 8

"Rectangle"

Column 9

"Rectangle"

Column 10

"Carré"

Column 11

"Carré"

Column 12

"Cercle"

Column 13

"Cercle"

Column 14

"Rectangle"

Column 15

"Carré"

Column 16

"Cercle"

Column 17

```
"Ellipse"  
Column 18  
"Ellipse"  
Column 19  
"Cercle"  
Column 20  
"Cercle"  
Column 21  
"Cercle"  
Column 22  
"Ellipse"
```

Détections de formes

```
clc; clear all; close all;  
  
% Charger l'image originale  
img = imread('formes.bmp');  
  
% Convertir en niveaux de gris  
img_gray = rgb2gray(img);  
  
% Ajouter du bruit gaussien à l'image  
img_bruit = imnoise(img_gray, 'gaussian', 0, 0.04);  
  
% Appliquer un filtrage bilatéral  
Dos = 2 * var(double(img_bruit(:)));  
img_filtree = imbilatfilt(img_bruit, Dos, 2);  
  
% Classifier les formes dans l'image bruitée  
resultats_bruit = classifierFormes(img_bruit);  
disp('Résultats de classification pour l''image bruitée :');
```

Résultats de classification pour l'image bruitée :

```
disp(resultats_bruit(1:min(50, length(resultats_bruit))));
```

```
Column 1  
"Cercle"  
Column 2  
"Cercle"
```

Column 3
"Cercle"
Column 4
"Cercle"
Column 5
"Cercle"
Column 6
"Cercle"
Column 7
"Cercle"
Column 8
"Cercle"
Column 9
"Cercle"
Column 10
"Cercle"
Column 11
"Cercle"
Column 12
"Cercle"
Column 13
"Cercle"
Column 14
"Cercle"
Column 15
"Cercle"
Column 16
"Cercle"
Column 17
"Cercle"
Column 18
"Cercle"
Column 19

"Cercle"

Column 20

"Cercle"

Column 21

"Cercle"

Column 22

"Cercle"

Column 23

"Cercle"

Column 24

"Cercle"

Column 25

"Cercle"

Column 26

"Cercle"

Column 27

"Cercle"

Column 28

"Cercle"

Column 29

"Cercle"

Column 30

"Cercle"

Column 31

"Cercle"

Column 32

"Cercle"

Column 33

"Cercle"

Column 34

"Cercle"

Column 35

```
"Cercle"  
Column 36  
"Cercle"  
Column 37  
"Cercle"  
Column 38  
"Cercle"  
Column 39  
"Cercle"  
Column 40  
"Cercle"  
Column 41  
"Cercle"  
Column 42  
"Cercle"  
Column 43  
"Cercle"  
Column 44  
"Cercle"  
Column 45  
"Cercle"  
Column 46  
"Cercle"  
Column 47  
"Cercle"  
Column 48  
"Cercle"  
Column 49  
"Cercle"  
Column 50  
"Cercle"
```

```
% Classifier les formes dans l'image filtrée bilatéralement  
resultats_filtree = classifierFormes(img_filtree);
```

```
disp('Résultats de classification pour l''image filtrée bilatéralement :');
```

Résultats de classification pour l'image filtrée bilatéralement :

```
disp(resultats_filtree);
```

Column 1

"Ellipse"

Column 2

"Cercle"

Column 3

"Rectangle"

Column 4

"Ellipse"

Column 5

"Ellipse"

Column 6

"Ellipse"

Column 7

"Cercle"

Column 8

"Ellipse"

Column 9

"Cercle"

Column 10

"Cercle"

Column 11

"Cercle"

Column 12

"Ellipse"

Column 13

"Ellipse"

Column 14

"Cercle"

Column 15

"Cercle"

Column 16

"Cercle"

Column 17

"Cercle"

Column 18

"Cercle"

Column 19

"Cercle"

Column 20

"Cercle"

Column 21

"Cercle"

Column 22

"Cercle"

Column 23

"Ellipse"

Column 24

"Cercle"

Column 25

"Cercle"

Column 26

"Cercle"

Column 27

"Cercle"

Column 28

"Cercle"

Column 29

"Ellipse"

Column 30

"Cercle"

Column 31

"Cercle"

Column 32
"Cercle"
Column 33
"Cercle"
Column 34
"Cercle"
Column 35
"Ellipse"
Column 36
"Cercle"
Column 37
"Cercle"
Column 38
"Cercle"
Column 39
"Cercle"
Column 40
"Cercle"
Column 41
"Ellipse"
Column 42
"Cercle"
Column 43
"Cercle"
Column 44
"Ellipse"
Column 45
"Cercle"
Column 46
"Cercle"
Column 47
"Cercle"

Column 48

"Cercle"

Column 49

"Cercle"

Column 50

"Cercle"

Column 51

"Ellipse"

Column 52

"Cercle"

Column 53

"Cercle"

Column 54

"Ellipse"

Column 55

"Ellipse"

Column 56

"Ellipse"

Column 57

"Ellipse"

Column 58

"Cercle"

Column 59

"Cercle"

Column 60

"Cercle"

Column 61

"Cercle"

Column 62

"Cercle"

Column 63

"Cercle"

Column 64

"Cercle"

Column 65

"Cercle"

Column 66

"Cercle"

Column 67

"Cercle"

Column 68

"Cercle"

Column 69

"Cercle"

Column 70

"Cercle"

Column 71

"Ellipse"

Column 72

"Cercle"

Column 73

"Cercle"

Column 74

"Cercle"

Column 75

"Cercle"

Column 76

"Cercle"

Column 77

"Rectangle"

Column 78

"Cercle"

Column 79

"Cercle"

Column 80

"Cercle"

Column 81

"Cercle"

Column 82

"Cercle"

Column 83

"Cercle"

Column 84

"Cercle"

Column 85

"Cercle"

Column 86

"Ellipse"

Column 87

"Ellipse"

Column 88

"Cercle"

Column 89

"Cercle"

Column 90

"Cercle"

Column 91

"Cercle"

Column 92

"Cercle"

Column 93

"Cercle"

Column 94

"Cercle"

Column 95

"Ellipse"

Column 96

"Ellipse"

Column 97
"Cercle"
Column 98
"Cercle"
Column 99
"Cercle"
Column 100
"Cercle"
Column 101
"Ellipse"
Column 102
"Cercle"
Column 103
"Ellipse"
Column 104
"Cercle"
Column 105
"Cercle"
Column 106
"Cercle"
Column 107
"Cercle"
Column 108
"Cercle"
Column 109
"Cercle"
Column 110
"Cercle"
Column 111
"Cercle"
Column 112
"Cercle"

Column 113
"Cercle"
Column 114
"Cercle"
Column 115
"Rectangle"
Column 116
"Carré"
Column 117
"Rectangle"
Column 118
"Carré"
Column 119
"Cercle"
Column 120
"Cercle"
Column 121
"Cercle"
Column 122
"Cercle"
Column 123
"Rectangle"
Column 124
"Carré"
Column 125
"Rectangle"
Column 126
"Ellipse"
Column 127
"Ellipse"
Column 128
"Cercle"
Column 129

```

"Cercle"
Column 130
"Cercle"
Column 131
"Ellipse"
Column 132
"Cercle"

% Classifier les formes dans l'image originale
resultats_originale = classifierFormes(img_gray);
disp('Résultats de classification pour l''image originale :');

Résultats de classification pour l'image originale :
disp(resultats_originale);

Column 1
"Ellipse"
Column 2
"Ellipse"
Column 3
"Ellipse"
Column 4
"Ellipse"
Column 5
"Ellipse"
Column 6
"Cercle"
Column 7
"Ellipse"
Column 8
"Rectangle"
Column 9
"Rectangle"
Column 10
"Carré"
Column 11

```

```
"Carré"  
Column 12  
"Cercle"  
Column 13  
"Cercle"  
Column 14  
"Rectangle"  
Column 15  
"Carré"  
Column 16  
"Cercle"  
Column 17  
"Ellipse"  
Column 18  
"Ellipse"  
Column 19  
"Cercle"  
Column 20  
"Cercle"  
Column 21  
"Cercle"  
Column 22  
"Ellipse"
```

Le test en question montre que la difficulté de détecter les formes augmente lorsque l'image présente une certaine forme de bruit. Dans ce cas, le code qui détecte les formes doit être affiné pour pouvoir fonctionner même en présence de bruit.

Détection des caractères

Dans une étude sur la conversion d'informations visuelles en données textuelles codées, deux approches algorithmiques de reconnaissance optique de caractères (OCR) ont été mises en œuvre pour traiter et analyser des images contenant du texte. La première approche se concentrait sur l'extraction d'un caractère unique à partir d'une image spécifique, tandis que la seconde visait à identifier et classifier une séquence complète de caractères au sein d'une image plus complexe et hétérogène. Les méthodes employées comprenaient

l'amélioration du contraste, la réduction du bruit et l'ajustement des seuils de binarisation pour optimiser l'identification des caractères. Les résultats démontrent la capacité de l'OCR à interpréter avec précision le contenu textuel dans divers contextes visuels, soulignant ainsi son potentiel pour automatiser la saisie de données et enrichir les interfaces utilisateur.

```
clc; clear all; close all;

% Chemin de l'image à analyser
cheminImage = 'LettreU.jpg'; % Assurez-vous que le chemin correspond à l'emplacement de votre fichier

% Image analysée
figure; imshow(cheminImage), title('Image analysée');

% Appeler la fonction pour reconnaître les caractères dans l'image
texteReconnu = reconnaîtreCaractères(cheminImage);

% Afficher le texte reconnu
disp(['Texte reconnu : ', texteReconnu]);
```

Texte reconnu : U

```
clc; clear all; close all;

% Chemin de l'image à analyser
cheminImage = 'Alphabet.jpg'; % Remplacez par le chemin réel de l'image

% Image analysée
figure; imshow(cheminImage), title('Image analysée');
```

Image analysée



```
% Appeler la fonction pour reconnaître tous les caractères dans l'image  
texteReconnu = reconnaîtreTousCaractères(cheminImage);
```

```
% Afficher le texte reconnu  
disp(['Texte reconnu : ', texteReconnu]);
```

```
Texte reconnu : ail b l mu  
I  
mg E
```

Lors de l'analyse d'une image présentant un caractère isolé, l'algorithme OCR a correctement détecté la présence d'un caractère mais a attribué par défaut une lettre majuscule "U" au lieu d'une minuscule "u". Cette tendance à la sur-généralisation vers les majuscules peut être attribuée aux contraintes inhérentes à la méthode OCR qui ne distingue pas finement les nuances de casse sans formation ou paramétrage spécifique. Par ailleurs, la reconnaissance des caractères dans une image avec un ensemble diversifié de lettres a échoué à délimiter et à interpréter chaque symbole individuellement, ce qui suggère des difficultés dans le traitement des variations de couleur, de taille et de contexte spatial des caractères. Ces observations mettent en lumière le besoin d'ajustements algorithmiques et de prétraitement d'image pour améliorer la précision de l'OCR dans des applications pratiques.