Computer vision

RAPPEL

- **La vision par ordinateur** se concentre sur la compréhension et l'interprétation des contenus visuels par les ordinateurs, en permettant aux machines de "voir" et de comprendre leur environnement de la même manière que le fait un être humain.
- **Le traitement d'images**, quant à lui, se concentre principalement sur la manipulation des images pour améliorer leur qualité, extraire des informations spécifiques ou effectuer des opérations telles que la restauration, la segmentation oula détection de contours
- **Une image**, fondamentalement, est représentation visuelle d'un objet illuminé par une source de lumière.
- **Une image** est essentiellement une matrice de pixels, où chaque pixel détient une valeur représentative de son niveau de luminosité ou de couleur.
- **Une image RGB** est un type d'image numérique où chaque pixel est représenté par trois valeurs correspondant aux composantes de couleur rouge (R), verte (G) et bleue (B).
- **Une image en niveaux de gris** est un type d'image où chaque pixel est représenté par une seule valeur de luminosité, généralement sur une échelle de 0 à 255. Cette valeur indique l'intensité lumineuse du pixel, où 0 représente le noir absolu et 255 représente le blanc absolu.
- **Une image indexée** est un type spécifique d'image numérique où les couleurs sont sélectionnées à partir d'une palette prédéfinie, plutôt que d'être représentées par des valeurs RVB (rouge, vert, bleu) pour chaque pixel individuel. Au lieu de cela, chaque pixel de l'image est associé à un index correspondant à une couleur dans une table de couleurs prédéfinie, également appelée palette de couleurs ou table de couleur.

COMMANDE

imread('cameraman.tif') : Cette fonction charge l'image appelée cameraman.tif et la stocke dans une variable appelée image_gray.
imfinfo('cameraman.tif') : Cette fonction récupère les informations détaillées sur l'image cameraman.tif et les stocke dans une variable appelée info_gray.
figure : Cette fonction crée une nouvelle fenêtre pour afficher l'image. Toutes les images affichées ensuite apparaîtront dans cette fenêtre jusqu'à ce qu'une nouvelle soit créée ou que le programme se termine.
imshow(image_gray) : Cette fonction affiche l'image stockée dans la variable image_gray dans la fenêtre graphique actuelle.
title('Image en niveaux de gris') : Cette fonction ajoute un titre à l'image affichée, indiquant qu'il s'agit d'une image en noir et blanc (niveaux de gris).
disp('Informations sur l''image en niveaux de gris:'); : Cette fonction affiche un message dans la console MATLAB pour introduire les informations sur l'image.
disp(info_gray); : Cette fonction affiche les informations détaillées de l'image, comme sa taille, son type et d'autres propriétés.
[image_indexed, map] = imread('corn.tif', 1);

- Cette fonction charge une image appelée corn.tif sous forme d'image indexée.
- L'image est stockée dans la variable image_indexed.
- La carte de couleurs associée (qui définit les couleurs utilisées dans l'image) est stockée dans la variable map.

imread('corn.tif', 1)

- Le chiffre 1 indique à MATLAB de charger la première image du fichier corn.tif.
- Si corn.tif contient plusieurs images, chaque image a un numéro (index) qui commence à 1 pour la première, 2 pour la deuxième, etc.
- Si le fichier ne contient qu'une seule image, ajouter 1 ne change rien, mais c'est une bonne habitude pour préciser l'image à charger.

Extraction des couleurs de l'image :

- Red = J(:,:,1); → Extrait le canal Rouge de l'image J.
- Green = J(:,:,2); → Extrait le canal Vert de l'image J.
- Blue = J(:,:,3); → Extrait le canal Bleu de l'image J.

En MATLAB, une image couleur est stockée sous forme d'une matrice 3D :

- La première couche (1) contient le rouge.
- La deuxième couche (2) contient le vert.
- La troisième couche (3) contient le bleu.

Affichage des images :

On divise la fenêtre en une grille 2x2 pour afficher l'image originale et ses trois canaux.
1 figure, subplot(2,2,1), imshow(J), title('image originale')
→ Affiche l'image originale dans la première case avec le titre "image originale".
2 subplot(2,2,2), imshow(Red), title('Canal Rouge')
→ Affiche le canal Rouge dans la deuxième case avec le titre "Canal Rouge".
3 subplot(2,2,3), imshow(Green), title('Canal Vert')
→ Affiche le canal Vert dans la troisième case avec le titre "Canal Vert".
4 subplot(2,2,4), imshow(Blue), title('Canal Bleu')
→ Affiche le canal Bleu dans la quatrième case avec le titre "Canal Bleu".

COMMANDE

subplot(2,2,1) :

Cette commande est utilisée pour diviser une figure en plusieurs parties et sélectionner une sous-fenêtre où une image ou un graphique sera affiché.

Explication :

- subplot(m, n, p) signifie :
  - m : Nombre total de lignes dans la grille.
  - n : Nombre total de colonnes dans la grille.
  - p : Position de la sous-fenêtre où le prochain affichage sera placé.

Dans subplot(2,2,1) :

- La figure est divisée en 2 lignes et 2 colonnes → soit 4 zones (2 × 2).
- Le chiffre 1 indique que l'affichage se fera dans la première case (en haut à gauche).

gray_image_rgb2gray = rgb2gray(image_color) :

- rgb2gray transforme une image couleur en image en niveaux de gris.
- Elle utilise la luminance des couleurs pour faire la conversion.
- image_color est l'image couleur d'origine.
- gray_image_rgb2gray est l'image obtenue après conversion en noir et blanc.

gray_image_manual = 0.2989 * Red + 0.5870 * Green + 0.1140 * Blue:

👉 Cette ligne convertit une image couleur en niveaux de gris en appliquant une formule mathématique.

💡 Pourquoi ces nombres ?

L'œil humain est plus sensible à certaines couleurs :

- Le vert (0.5870) influence le plus l'intensité lumineuse.
- Le rouge (0.2989) a un effet moyen.
- Le bleu (0.1140) est le moins visible pour l'œil humain.

🔹 Explication des éléments :

- Red, Green, et Blue sont les trois canaux de couleur extraits de l'image.
- La formule pondère chaque couleur en fonction de sa visibilité.
- gray_image_manual est l'image obtenue en niveaux de gris après la conversion.

y = ind2rgb(x, map);

- Convertit une image indexée x en image couleur RGB.
- map est la carte de couleurs associée à l'image indexée.
- La sortie y est une image en couleur (RGB).

y = ind2gray(x, map);

- Transforme une image indexée x en image en niveaux de gris.
- map (carte de couleurs) est utilisé pour la conversion.
- La sortie y est une image en noir et blanc (grayscale).

[y, map] = gray2ind(x, n);

- Convertit une image en niveaux de gris x en une image indexée.
- n est le nombre de couleurs dans la palette (map).
- La sortie y est l'image indexée et map est la carte de couleurs.

y = rgb2gray(x);

- Transforme une image couleur RGB x en image en niveaux de gris.
- L'intensité des couleurs est pondérée selon la sensibilité de l'œil humain.

La sortie y est une image en noir et blanc

COMMANDE

y = gray2rgb(x);

- Convertit une image en niveaux de gris x en une image RGB.
- Les trois canaux (R, G, B) contiennent la même valeur de gris pour chaque pixel.
- La sortie y est une image couleur (mais en nuances de gris).

[y, map] = rgb2ind(x, n);

- Convertit une image couleur RGB x en une image indexée.
- n est le nombre de couleurs dans la palette (map).
- La sortie y est l'image indexée et map est la carte de couleurs.

image()

- Affiche une matrice d'entrée comme une image.
- Cette fonction est utilisée pour afficher une matrice numérique comme une image, où les valeurs de la matrice déterminent les couleurs des pixels.
- 💡 Exemple : image(matrix);

imtool()

- Affiche une image avec des outils supplémentaires.
- Cette fonction permet de visualiser l'image avec des outils interactifs comme le zoom, les ajustements de contraste, etc.
- 💡 Exemple : imtool(image);

imagesc()

- Affiche une image avec interpolation des couleurs.
- Utilisée pour afficher des matrices où les couleurs sont automatiquement interpolées pour rendre l'image plus lisse et esthétique.
- 💡 Exemple : imagesc(matrix);

numel(I)

- Cette fonction compte le nombre total d'éléments dans la variable I (qu'il s'agisse d'une image, d'une matrice, etc.).
- Par exemple, pour une image, elle compte le nombre total de pixels (lignes × colonnes × canaux).

💡 Exemple :

```
I = imread('cameraman.tif'); % Charger une image
total_elements = numel(I); % Nombre total de pixels de l'imagedisp(total_elements);
```

whos I

- Affiche des informations détaillées sur la variable I, comme la taille, le type de données, et la mémoire utilisée.
- Cette commande est utile pour mieux comprendre les propriétés de la variable.

💡 Exemple :

```
I = imread('cameraman.tif');
whos I; % Affiche des informations sur l'image
```

imfinfo('image.png')

- Cette fonction fournit des informations détaillées sur l'image, telles que la taille, le format de fichier, et les détails sur les couleurs.
- Utile pour obtenir des informations métadonnées sur l'image.

💡 Exemple :

```
info = imfinfo('cameraman.tif'); % Afficher les informations détaillées
```

COMMANDE

size(I)

- Renvoie la taille de l'image (nombre de lignes, colonnes, et de canaux) dans la variable I.
- Utile pour connaître les dimensions d'une image.

💡 Exemple :

```
I = imread('cameraman.tif');
dimensions = size(I); % Dimensions de l'image (lignes, colonnes, canaux)disp(dimensions);
```

class(I)

- Cette fonction renvoie le type de données de la variable I.
- Par exemple, elle pourrait renvoyer 'uint8' pour une image en niveaux de gris ou 'double' pour des images traitées.

💡 Exemple :

```
I = imread('cameraman.tif');
data_type = class(I); % Type de données de l'imagedisp(data_type);
```

imhist(I)

- La fonction imhist() affiche l'histogramme d'une image. Un histogramme montre la répartition des intensités de pixels dans l'image (pour les images en niveaux de gris), c'est-à-dire la fréquence à laquelle chaque valeur de pixel (de 0 à 255 pour une image de type uint8) apparaît dans l'image.

Explication :

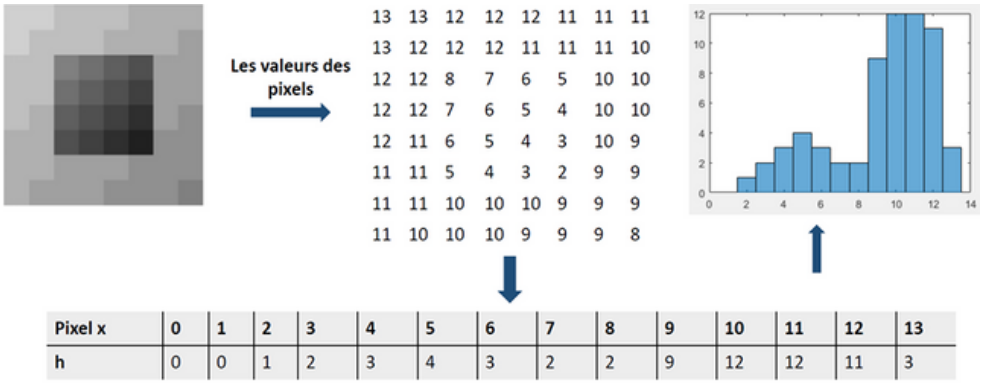
- L'axe des x représente les différentes valeurs de pixel (de 0 à 255).
- L'axe des y représente le nombre de pixels qui ont une valeur particulière.

💡 Exemple :

```
I = imread('cameraman.tif'); % Charger une image en niveaux de gris
imhist(I); % Afficher l'histogramme de l'image
title('Histogramme de l''image');
```

DEF

- **Un histogramme d'une image** est une représentation graphique de la distribution des valeurs des pixels dans cette image. C'est un outil essentiel en traitement d'images pour analyser les caractéristiques de luminosité et decontraste.



Pixel x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
h	0	0	1	2	3	4	3	2	2	9	12	12	11	3

Utilité de l'Histogramme : L'histogramme est un outil précieux pour :

- **Analyser le Contraste :** Une image avec un faible contraste aura un histogramme concentré dans une petite gamme de valeurs, tandis qu'une image avec un bon contraste aura un histogramme étendu sur toute la plage des valeurs.
- **Détecter la Luminosité :** Un histogramme décalé vers la gauche indique une image sombre, tandis qu'un histogramme décalé vers la droite indique une image lumineuse.
- **Équilibrer les Couleurs :** Pour les images en couleur, analyser les histogrammes des différents canaux permet de voir si une couleur est dominante ou si les couleurs sont équilibrées

Pour calculer et afficher l'histogramme d'une image sur matlab , on utilise la fonction **imhist**

TRAITEMENTS BASIQUES D'IMAGE

Expansion Dynamique/ recadrage dynamique

L'expansion dynamique est une technique de traitement d'image qui améliore le contraste d'une image en élargissant la plage des niveaux de gris ou des couleurs. Cela permet de rendre les détails de l'image plus visibles, surtout si l'image originale a une plage de niveaux de gris restreinte.

Comment ça fonctionne ?

- Avant l'expansion : Si l'image a des valeurs de pixels concentrées dans une petite plage (par exemple, entre 50 et 180), les différences entre les pixels peuvent être difficiles à voir.
- Après l'expansion : Cette petite plage de valeurs est étirée pour utiliser toute la plage possible, c'est-à-dire de 0 à 255. Cela permet de mieux distinguer les détails de l'image, car les intensités sont mieux réparties sur toute la gamme de couleurs.

Exemple :

Imaginons que vous avez une image en niveaux de gris où les valeurs des pixels vont de 50 à 180. L'expansion dynamique va étirer ces valeurs pour qu'elles couvrent toute la plage de 0 à 255.

Principe de l'Expansion Dynamique :

1. Trouver les valeurs minimales et maximales des pixels dans l'image.
2. Étendre ces valeurs pour qu'elles couvrent toute la plage possible, c'est-à-dire de 0 à 255.
3. Cela augmente le contraste, surtout dans les zones où il était difficile de distinguer les différences de niveaux de gris.

TRAITEMENTS BASIQUES D'IMAGE

La fonction rescale(img)

La fonction rescale est utilisée pour réajuster les valeurs d'une matrice ou d'un tableau afin qu'elles se situent entre deux valeurs spécifiées. Par défaut, elle redimensionne les valeurs pour qu'elles soient entre 0 et 1. Cela permet de mieux contrôler la plage des valeurs de l'image ou de la matrice.

La fonction imadjust(img)

La fonction imadjust en MATLAB est utilisée pour ajuster les niveaux d'intensité d'une image. Elle permet d'améliorer le contraste de l'image en manipulant la gamme de valeurs des pixels.

Comment fonctionne imadjust ?

- Cette fonction ajuste les valeurs des pixels d'une image afin d'améliorer la visibilité des détails dans des zones où les pixels étaient proches les uns des autres.
- Par exemple, si une image a des pixels concentrés dans une plage étroite, imadjust peut étirer cette plage pour augmenter le contraste.

La fonction histeq(img)

La fonction histeq en MATLAB applique une technique d'égalisation d'histogramme à une image, ce qui permet d'améliorer son contraste. Cette méthode redistribue les intensités des pixels pour rendre l'histogramme de l'image aussi uniforme que possible sur toute la plage dynamique des pixels (généralement de 0 à 255 pour une image en niveaux de gris).

Comment ça fonctionne ?

- L'objectif de l'égalisation d'histogramme est de répartir uniformément les intensités de pixels sur toute la gamme des valeurs possibles. Cela permet de mieux distinguer les détails dans les zones où les pixels étaient auparavant concentrés dans une petite plage d'intensité.
- L'égalisation augmente le contraste, surtout dans les images où certaines zones sont trop sombres ou trop claires pour qu'on puisse voir les détails.

TRAITEMENTS BASIQUES D'IMAGE

Egalisation adaptative

imbinarize(img)

La fonction imbinarize(img) en MATLAB sert à convertir une image en niveaux de gris (comme une photo noir et blanc) en image binaire. Cela signifie qu'elle transforme chaque pixel de l'image en noir ou blanc seulement, en fonction de son intensité.

Comment ça fonctionne ?

- Imaginez que vous avez une photo avec des pixels clairs (gris clair ou blanc) et des pixels sombres (gris foncé ou noir).
- La fonction imbinarize analyse l'image et transforme chaque pixel en noir ou blanc en fonction d'un seuil de luminosité.
- Par exemple, les pixels plus sombres deviendront noirs, et les pixels plus clairs deviendront blancs.

Pourquoi utiliser imbinarize ?

- Cela permet de simplifier l'image, ce qui peut être utile pour faire ressortir des éléments importants, comme du texte ou des objets.
- C'est une méthode courante utilisée dans des tâches comme la reconnaissance de texte ou la détection d'objets.

Exemple :

```
img = imread('photo_gris.png');  % Charger une image en niveaux de gris
img_bin = imbinarize(img);       % Convertir l'image en image binaire
imshow(img_bin);                 % Afficher l'image binaire
```

TRAITEMENTS BASIQUES D'IMAGE

Le seuillage (Thresholding)

Le seuillage est une technique utilisée pour convertir une image en niveaux de gris en une image binaire (noir et blanc). Cela consiste à choisir un seuil, c'est-à-dire une valeur de luminosité, pour distinguer les pixels sombres des pixels clairs.

1. Seuillage manuel :

Dans le seuillage manuel, vous choisissez un seuil de manière manuelle pour décider quels pixels deviendront noirs et quels pixels deviendront blancs. Si un pixel est plus lumineux que ce seuil, il devient blanc; s'il est plus sombre, il devient noir.

Exemple de seuillage manuel :

```
seuil = 0.5;                % Choisir un seuil (entre 0 et 1)
img_bin = imbinarize(img, seuil); % Appliquer le seuillage manuel
```

- Ici, 0.5 signifie que les pixels dont la valeur est supérieure à 0.5 seront blancs, et ceux dont la valeur est inférieure seront noirs.
- Le seuil peut être choisi entre 0 (noir) et 1 (blanc), où 0.5 est la valeur intermédiaire.

2. Seuillage automatique :

Dans le seuillage automatique, la fonction choisit un seuil automatiquement en fonction des caractéristiques de l'image. Cela est particulièrement utile si vous n'êtes pas sûr du bon seuil à utiliser.

La fonction graythresh en MATLAB calcule automatiquement ce seuil en utilisant une méthode appelée Otsu's method, qui maximise la séparation entre les pixels sombres et clairs de l'image.

Exemple de seuillage automatique avec graythresh :

```
seuil = graythresh(img);     % Calculer le seuil automatique
img_bin = imbinarize(img, seuil); % Appliquer le seuillage automatique
imshow(img_bin);             % Afficher l'image binaire
```

- graythresh analyse l'image et détermine un seuil optimal pour séparer les pixels sombres et clairs.
- Le seuil retourné par graythresh est une valeur entre 0 et 1.
- Par exemple, si graythresh retourne 0.6, cela signifie que tous les pixels avec une intensité supérieure à 0.6 seront blancs, et ceux avec une intensité inférieure seront noirs.

FILTRAGE D'IMAGES

Le filtrage d'images est une méthode utilisée pour améliorer une image. Il aide à enlever les bruits (informations inutiles ou perturbations) et à mettre en évidence les détails importants. En gros, on applique une sorte de filtre à l'image pour la rendre plus nette et plus claire, comme si on passait l'image au travers d'un tamis pour retirer les impuretés.

Sources de dégradation des images :

Les images peuvent se détériorer (ou devenir de moins bonne qualité) pour plusieurs raisons, comme par exemple :

1. Bruit :
    - Le bruit est quand il y a des variations aléatoires de couleurs ou de lumière dans l'image. Cela peut arriver à cause de mauvaises conditions de prise de vue, comme une faible luminosité ou un capteur de caméra de mauvaise qualité. Le bruit rend l'image floue ou de mauvaise qualité.
2. Flou :
    - Le flou survient souvent quand l'appareil photo bouge pendant que la photo est prise, ou si l'objet que vous photographiez est en mouvement. L'image devient floue et il est plus difficile de voir les détails.
3. Distorsion :
    - Parfois, les lentilles de la caméra peuvent déformer l'image. Cela peut provoquer des lignes droites qui apparaissent courbées ou d'autres effets qui rendent l'image moins nette. Cela est souvent dû à des imperfections dans l'équipement de la caméra.

FILTRAGE D'IMAGES

COMMANDE

- **J = imnoise(I, 'gaussian')**
- → Ajoute du bruit gaussien (comme un effet de grain) avec une moyenne de 0 et une petite variation.
- **J = imnoise(I, 'gaussian', m)**
- → Ajoute du bruit gaussien avec une moyenne personnalisée (m), mais toujours une petite variation.
- **J = imnoise(I, 'gaussian', m, var_gauss)**
- → Ajoute du bruit gaussien avec une moyenne (m) et une variation spécifique (var_gauss).
- **J = imnoise(I, 'poisson')**
- → Ajoute un bruit de type "Poisson", qui dépend des valeurs de l'image.
- **J = imnoise(I, 'salt & pepper')**
- → Ajoute du bruit "sel et poivre" (des pixels blancs et noirs aléatoires). Par défaut, environ 5% des pixels sont affectés.
- **J = imnoise(I, 'salt & pepper', d)**
- → Même chose que le précédent, mais en ajustant la quantité de bruit avec (d).
- **J = imnoise(I, 'speckle')**
- → Ajoute du bruit "speckle" (petits points aléatoires) avec une variation par défaut de 0,05.
- **J = imnoise(I, 'speckle', var_speckle)**
- → Ajoute du bruit "speckle" avec une variation personnalisée (var_speckle).

FILTRAGE D'IMAGES

CONVOLUTION

La convolution est une opération mathématique qui combine deux ensembles d'informations. En traitement d'image, la convolution est utilisée pour appliquer des filtres à une image. Un filtre est une matrice (ou noyau) que l'on fait glisser sur l'image pour transformer ses pixels.

CONVOLUTION D’IMAGES

Une matrice de convolution (appelée aussi noyau) est une petite grille de nombres utilisée pour modifier une image. Elle sert à appliquer différents effets comme :

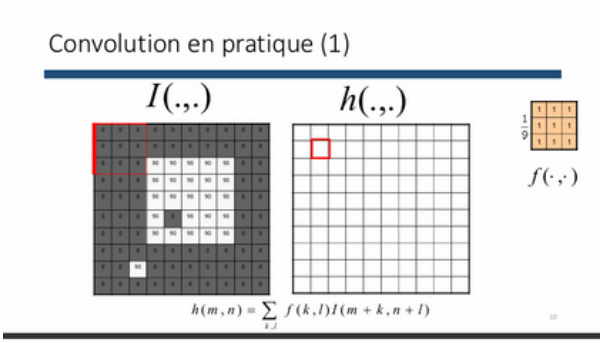
- Flouter l’image
- Rendre plus nette
- Détecter les contours
- Créer un effet de relief

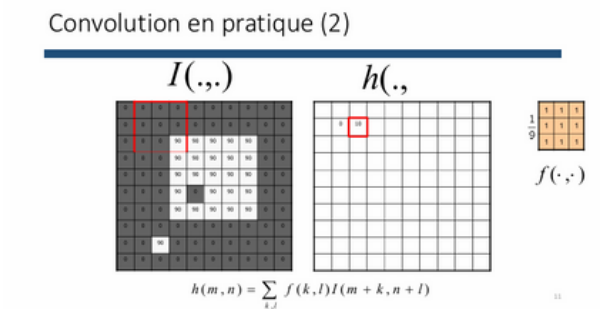
FILTRAGE D'IMAGES

CONVOLUTION PRATIQUE

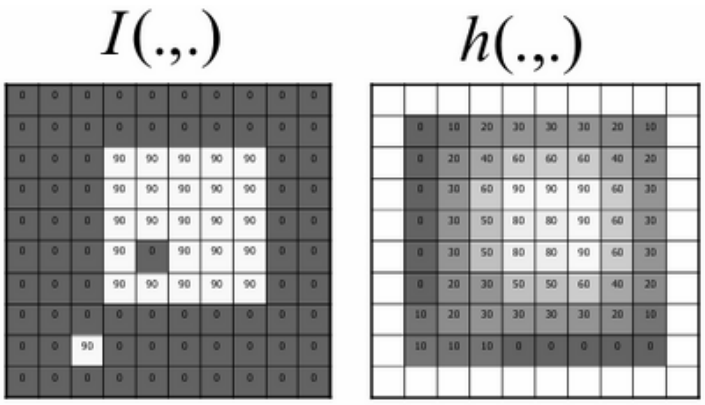
Comment ça marche ?

1. L'image d'origine (à gauche) contient des valeurs de pixels.
2. Le noyau (ou filtre) (en haut à droite) est une petite matrice qui sert à modifier l'image.
3. On applique le noyau sur chaque partie de l'image :
   - On place le noyau sur une zone de l'image (en rouge).
   - On multiplie chaque valeur de l'image par la valeur correspondante du noyau.
   - On additionne le tout et on met le résultat dans l'image de sortie (à droite).
4. On répète l'opération sur toute l'image, en déplaçant le noyau.





RESULTAT



FILTRAGE D'IMAGES

FILTRES MOYENNEURS ET GAUSSIENS

FILTRES MOYENNEURS :

Les filtres moyenneurs sont utilisés pour adoucir une image en prenant la moyenne des pixels voisins. Cela aide à réduire le bruit, mais cela peut aussi rendre l'image un peu floue, surtout autour des bords. Le filtre prend chaque pixel et remplace sa valeur par la moyenne des valeurs de ses voisins, créant ainsi un effet de lissage.

Avantages : Réduit le bruit dans l'image.

Inconvénients : Peut flouter les bords et détails importants.

Commande MATLAB :

fspecial('average', taille) : Crée un filtre moyenneur avec une taille spécifique (par exemple 3x3).

imfilter(image, filtre) : Applique le filtre à l'image.

```
filtre_moyenneur = fspecial('average', [3 3]);
image_filtrée = imfilter(image, filtre_moyenneur);
```

FILTRAGE D'IMAGES

FILTRES MOYENNEURS ET GAUSSIENS

FILTRES GAUSSIENS :

Les filtres gaussiens utilisent une fonction mathématique appelée la courbe gaussienne pour appliquer un lissage plus naturel. Au lieu de simplement prendre la moyenne, ce filtre donne plus de poids aux pixels proches du centre et moins de poids à ceux plus éloignés. Cela permet de conserver de meilleurs détails et bords tout en réduisant le bruit.

- **Avantages :** Conserve mieux les détails et bords de l'image tout en réduisant le bruit.
- **Inconvénients :** Moins agressif que le filtre moyenneur pour supprimer le bruit.

Commande MATLAB :

- fspecial('gaussian', taille, écart-type) : Crée un filtre gaussien avec une taille et un écart-type spécifiques.
- imfilter(image, filtre) : Applique le filtre gaussien à l'image.

```
filtre_gaussien = fspecial('gaussian', [3 3], 0.5);
image_filtrée = imfilter(image, filtre_gaussien);
```

FILTRAGE D'IMAGES

FILTRE MÉDIAN

Le filtre médian est une technique de filtrage efficace pour éliminer le bruit de type "sel et poivre", qui se caractérise par des pixels très clairs ou très sombres répartis de manière aléatoire sur l'image. Contrairement aux filtres moyenneurs qui utilisent la moyenne des voisins, le filtre médian remplace chaque pixel par la valeur médiane de ses voisins. Cela permet de préserver mieux les détails de l'image tout en supprimant efficacement le bruit, notamment dans les zones où il y a des pixels anormalement clairs ou sombres.

Avantages :

- Efficace pour éliminer le bruit "sel et poivre".
- Préserve les bords et les détails mieux que les filtres moyenneurs.
- Conserve les objets dans l'image tout en supprimant les bruits.

Inconvénients :

- Peut flouter légèrement les images avec des détails fins.

Commande MATLAB :

- medfilt2 : Applique un filtre médian à une image 2D.

```
image_filtrée = medfilt2(image, [3 3]);
```

FILTRAGE D'IMAGES

COMMANDE

immse (Mean Squared Error) :

La fonction immse (Mean Squared Error, ou Erreur Quadratique Moyenne en français) est utilisée pour mesurer la différence entre deux images. C'est une méthode couramment utilisée pour évaluer la qualité d'une image, en comparant une image originale avec une image modifiée ou filtrée.

Comment ça fonctionne ?

- immse compare chaque pixel de l'image de référence (par exemple, l'image originale) avec l'image traitée ou reconstruite.
- Elle calcule la moyenne des carrés des différences entre les pixels correspondants des deux images.
- Plus l'erreur est faible, plus les deux images sont similaires. Si l'erreur est élevée, cela signifie que les images sont différentes.

Commande :

```
erreur = immse(image1, image2);
```

- image1 et image2 sont les deux images à comparer.
- erreur contient la valeur de l'erreur quadratique moyenne entre les deux images. Plus cette valeur est faible, plus les images sont proches.

Exemple :

```
image1 = imread('image_originale.png');
image2 = imread('image_modifiee.png');
erreur = immse(image1, image2);
```

FILTRAGE D'IMAGES

COMMANDE

brisque (Blind/Referenceless Image Spatial Quality Evaluator) :

La fonction brisque est utilisée pour mesurer la qualité d'une image sans avoir besoin de la comparer à une image de référence (d'où le terme "blind" ou "sans référence"). Contrairement à immse qui nécessite une image de référence, brisque analyse l'image seule et évalue sa qualité de manière objective en fonction de plusieurs critères.

Comment ça fonctionne ?

- brisque évalue la qualité d'une image en fonction de l'aspect visuel de l'image. Cela inclut des aspects comme le bruit, la distorsion, le flou, etc.
- Elle renvoie une valeur de qualité, où une valeur plus basse indique une meilleure qualité (moins de distorsion ou de bruit).

Commande :

```
qualite = brisque(image);
```

- image est l'image dont on veut évaluer la qualité.
- qualite est un score de qualité basé sur l'évaluation de l'image. Plus la valeur est faible, meilleure est la qualité de l'image.

Exemple :

```
image = imread('image_a_evaluer.png');
qualite = brisque(image);
disp(qualite);
```

Cela affiche la qualité de l'image sans avoir besoin d'une image de référence.

SEGMENTATION D'UNE IMAGE TEXTURÉE

Objectif:

Nous allons explorer une méthode avancée de segmentation d'image pour séparer un objet de son arrière-plan. Le défi réside dans le fait que l'objet et l'arrière-plan sont texturés avec les mêmes couleurs, rendant la binarisation traditionnelle inefficace. Nous utiliserons des fonctions de filtrage de texture et d'autres traitements pour parvenir à une segmentationprécise.
L'objectif est de segmenter l'image de manière que l'objet (le poisson) apparaisse en blanc sur un fond noir.

SEGMENTATION D'UNE IMAGE TEXTURÉE

COMMANDE

rangefilt()

La fonction rangefilt est utilisée en traitement d'image pour mesurer la différence entre le pixel le plus clair et le pixel le plus foncé dans une petite région autour de chaque pixel d'une image.

Comment ça fonctionne ?

- Pour chaque pixel, MATLAB regarde une petite fenêtre autour de lui (généralement de taille 3x3).
- Il trouve la valeur maximale et la valeur minimale des pixels dans cette fenêtre.
- Il soustrait la valeur minimale de la valeur maximale pour obtenir la plage (range).
- Il remplace ensuite le pixel central par cette valeur.

Pourquoi l'utiliser ?

- Permet de détecter les bords et les zones de contraste élevé dans une image.
- Met en évidence les détails fins et les textures.
- Peut être utilisé pour repérer les changements brusques dans une image.

Commande MATLAB :

```
image = imread('image.png'); % Charger l'image
resultat = rangefilt(image); % Appliquer le filtre de plage
imshow(resultat); % Afficher le résultat
```

SEGMENTATION D'UNE IMAGE TEXTURÉE

COMMANDE

- **imcomplement() - Inversion des pixels**

Objectif : Inverser l'image binaire (changer les 1 en 0 et les 0 en 1).
Utilisation : On transforme une image où l'objet est en noir sur un fond blanc, en une image où l'objet devient blanc sur un fond noir.
Commande :

```
image_inverse = imcomplement(image_binaire);
```

- **imclearborder() - Suppression des objets collés à la bordure**

Objectif : Supprimer les objets qui touchent les bords de l'image.
Pourquoi ? Ces objets peuvent être du bruit ou des parties incomplètes d'un objet réel.
Commande :

```
image_sans_bord = imclearborder(image_binaire);
```

- **bwareaopen() - Suppression des petits objets**

Objectif : Enlever les objets trop petits (moins qu'un certain nombre de pixels).
Pourquoi ? Pour supprimer du bruit ou des détails insignifiants dans l'image.
Commande (ex : suppression des objets de moins de 50 pixels) :

```
image_filtre = bwareaopen(image_binaire, 50);
```

- **strel() et imclose() - Fermeture morphologique**

Objectif : Combler les trous et connecter les parties d'un objet.
Pourquoi ? Si un objet est formé de plusieurs parties séparées par de petits espaces, cette opération les relie.
Commande :

```
se = strel('disk', 6); % Élément structurant en forme de disque (rayon = 6)
image_fermee = imclose(image_binaire, se);
```

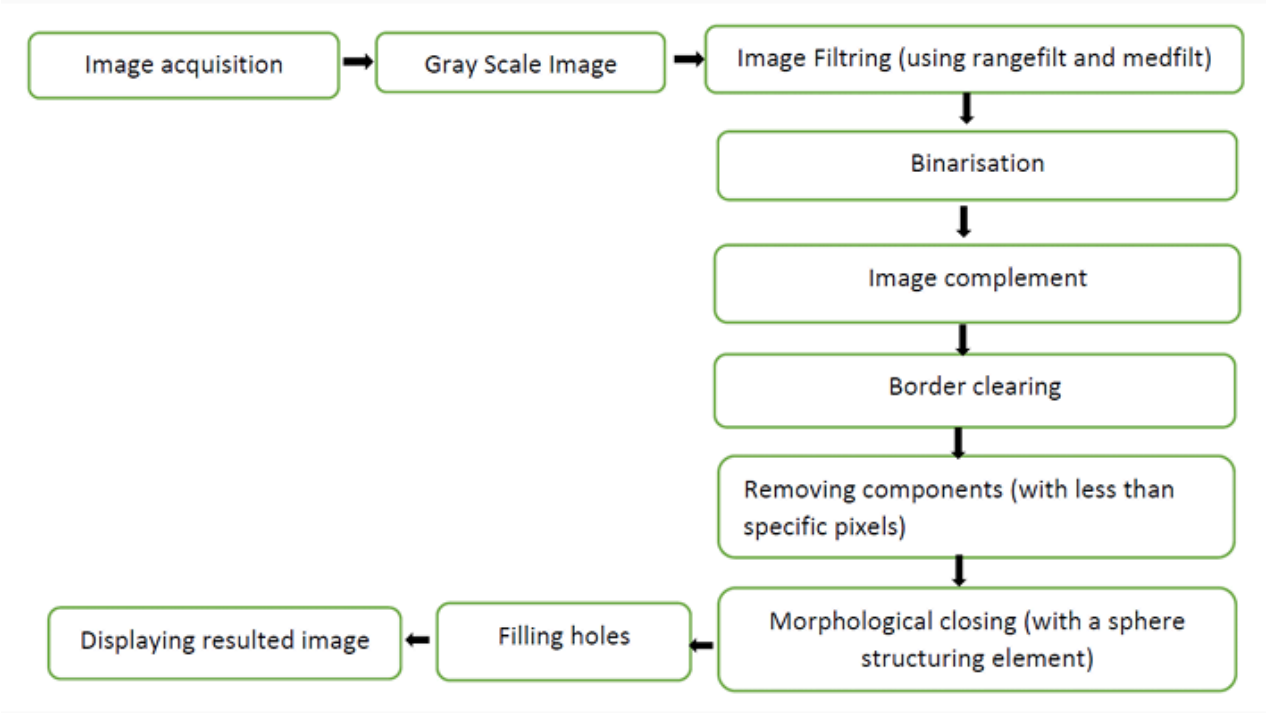
- **imfill() - Remplissage des trous**

Objectif : Remplir les trous à l'intérieur des objets.
Pourquoi ? Certains objets peuvent être creux après binarisation, et cette fonction les rend pleins.
Commande :

```
image_finale = imfill(image_binaire, 'holes');
```

SEGMENTATION D'UNE IMAGE TEXTURÉE

PROCESSUS



SEGMENTATION D'UNE IMAGE TEXTURÉE

Détection du Contour Horizontal d’une Image

Pour détecter les contours horizontaux dans une image, on utilise généralement des filtres de dérivation tels que les filtres Sobel ou de Prewitt, Ces filtres mettent en évidence les changements d’intensité dans la direction horizontale.

🔹 Pourquoi utiliser ces filtres ?

Les filtres de Sobel et de Prewitt permettent de repérer les changements d'intensité dans une image, c'est-à-dire les endroits où la couleur ou la luminosité varient brusquement. Cela aide à identifier les contours, comme les bords d'un objet.

🔹 Comment fonctionnent ces filtres ?

Ils utilisent des matrices (des petits tableaux de nombres) qui sont appliquées sur l'image pour détecter les zones de transition entre les couleurs.

📌 Filtre de Sobel (Horizontal)

- Plus précis, il met mieux en valeur les contours lisses.
- Il est souvent utilisé pour des détections avancées des bords.

📌 Filtre de Prewitt (Horizontal)

- Plus simple que Sobel, mais moins précis.
- Il est utilisé pour des détections de contours basiques.

Filtre de Sobel (Horizontal)

Le filtre de Sobel pour la détection des contours horizontaux est le suivant :

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Filtre de Prewitt (Horizontal)

Le filtre de Prewitt pour la détection des contours horizontaux est similaire mais plus simple :

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

SEGMENTATION D'UNE IMAGE TEXTURÉE

Détection du Contour Vertical d’une Image

Pour détecter les contours verticaux dans une image, on utilise généralement des filtres de dérivation tels que les filtres Sobel ou de Prewitt, mais orientes pour capturer les changement d’intensité dans la direction verticale.

🔹 Pourquoi utiliser ces filtres ?

Ces filtres permettent de repérer les changements d'intensité dans une image de gauche à droite. Cela aide à détecter les contours verticaux, comme les bords d'un immeuble ou d'un arbre sur une photo.

🔹 Comment fonctionnent ces filtres ?

Ils utilisent des matrices appliquées sur l’image pour mettre en évidence les transitions verticales.

📌 Filtre de Sobel (Vertical)

- Plus précis, il met mieux en valeur les contours verticaux en les rendant plus lisses.
- Utilisé pour des analyses avancées des bords.

📌 Filtre de Prewitt (Vertical)

- Plus simple que Sobel, mais moins précis.
- Utilisé pour une détection basique des contours verticaux.

Filtre de Sobel (Vertical)

Le filtre de Sobel pour la détection des contours verticaux est le suivant :

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Filtre de Prewitt (Vertical)

Le filtre de Prewitt pour la détection des contours verticaux est similaire mais plus simple :

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

SEGMENTATION D'UNE IMAGE TEXTURÉE

Filtre de Laplacien

🔹 Qu’est-ce que le filtre de Laplacien ?

- Il met en évidence les zones où la luminosité change brusquement, ce qui correspond aux bords des objets.
- Contrairement aux filtres de Sobel et de Prewitt (qui détectent les bords soit horizontalement, soit verticalement), le filtre de Laplacien détecte les contours dans toutes les directions en même temps.

🔹 Comment fonctionne-t-il ?

Le filtre de Laplacien utilise une matrice de convolution (appelée noyau) qui, lorsqu’elle est appliquée à une image, accentue les variations soudaines de luminosité.

📌 Exemples de noyaux Laplaciens

1️⃣ Noyau Laplacien 3x3 sans centre négatif

- Ce noyau détecte les contours en accentuant les pixels centraux par rapport à leurs voisins.

2️⃣ Noyau Laplacien 3x3 avec centre négatif

- Ce noyau est une variante qui met encore plus en évidence les contours en appliquant un poids plus fort au centre.

🔹 Pourquoi utiliser le filtre de Laplacien ?

- Il est simple et efficace pour mettre en valeur les bords d’une image.
- Il est souvent utilisé en combinaison avec d’autres filtres pour améliorer la détection des contours.

1. **Noyau Laplacien 3x3 sans centre négatif :**

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

2. **Noyau Laplacien 3x3 avec centre négatif :**

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

MATRIC DES FILTRES

Filtre moyenneur

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Filtre Gaussien

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$