

### But

Aborder quelques méthode de segmentation d'image:

- Méthode de seuillage sur histogramme
- Méthode de croissance des régions
- Algorithme de k-means
- Méthode de division/fusion.

### 1. Segmentation par la méthode de seuillage d'histogramme

- Quant les régions dans l'image ont des niveaux de gris bien distincts, on peut aisément les distinguer sur l'histogramme (zones distinctes)
- et en calculant bien les seuils de séparations entre les zones d'histogrammes, on arrivera à séparer de ce fait les points des régions en question et de ce fait les segmenter.
- La méthode d'Otsu permet de calculer un ou plusieurs seuils de séparations entre les zones d'histogrammes, dits seuils de segmentation des régions dans l'image

### Les commandes Matlab

<code>thresh = multithresh(I,N);</code>	<ul style="list-style-type: none"><li>– permet de calculer, par la méthode d'Otsu, les N seuils de segmentation de l'image I.</li></ul> <p><b>thres</b>: serait un vecteur de taille <math>(1 \times N)</math></p>
<code>thresh = multithresh(I);</code>	<ul style="list-style-type: none"><li>– si on ne précise pas N, par défaut N=1, on aura un seul seuil de seuillage.</li></ul> <p><b>thres</b> : serait donc un scalaire</p>
<code>Is = imquantize(I, thres);</code>	<ul style="list-style-type: none"><li>– Quantifie (seuille) l'image I selon le seuil ou les seuils contenus dans thres.</li><li>– Pour chaque seuil<ul style="list-style-type: none"><li>o <math>I(x,y)=0</math> si <math>I(x,y)&lt;\text{seuil}</math> :</li><li>o <math>I(x,y)=\text{étiquette}</math> si <math>I(x,y)\geq\text{seuil}</math></li></ul></li><li>– Is : est une matrice d'étiquettes dont les valeurs (étiquettes) sont = 1,2,...,N</li></ul>
<code>Zrgb=label2rgb(Z_label);</code>	Permet de convertir une matrice d'étiquettes (labels) Z_label en une matrice couleur Zrgb.

### Manipulation

1. Lire l'image 'medtest.png', affectez-la à la variable I, puis afficher-là.
2. Afficher son histogramme. Distinguer les zones séparables sur histogramme.
3. calculer un seuil de segmentation ( $N = 1$ )

4. segmenter l'image I, pour obtenir la matrice  $I_s$ .
  - ouvrir la matrice  $I_s$ , et vérifier que son contenu est formé de 0 et 1
  - Afficher le résultat de segmentation
5. calculer deux seuils de segmentation ( $N = 2$ )
6. segmenter l'image I, pour obtenir la matrice étiquette  $I_s$ ,
  - ouvrir la matrice  $I_s$ , et vérifier que son contenu est formé de 0, 1 et 2
  - convertissez-la en une image couleur
  - Afficher le résultat de segmentation
  - commenter
7. Refaire la question 6. pour  $N = 3$ , puis 4

## 2. Segmentation par la méthode de croissance de région (growing region)

Pour segmenter une région  $R$  ayant un écart type de  $\sigma_R$

1. on commence par un point germe (seed) d'un niveau de gris donné (c'est la valeur initiale de la moyenne de  $R$ )
2. puis on croit la région autour du germe, en ajoutant à la région tous les point avoisinants dont les niveaux de gris  $I(x, y)$  vérifient la condition :
 
$$|I(x, y) - \mu_R| \leq \sigma_R, \quad \text{où } \mu_R \text{ est la moyenne de } R$$
3. Recalculer la moyenne de  $R$  après croissance
4. Répéter les opérations 2 et 3, jusqu'à ce qu'il n'y ait plus de croissance.

### Les Commandes Matlab :

```
J = regiongrowing(I,x,y,sigma_R);
```

Permet d'appliquer l'algorithme de croissance **d'une seule région** :

- dans l'image I,
- autour du germe (pixel initial) de coordonnées  $[x,y]$ ,
- l'écart type de la région  $\sigma_R$
- J a la même taille que I, c'est une matrice d'étiquettes (labels) et dont les valeurs sont :
  - 1 (true) : point segmenté
  - 0 (false) : point non-segmenté

```
[x, y] = getpts;
```

Permet d'extraire les coordonnées du point de la figure pointé avec la souris

### Manipulation

1. Lire l'image 'medtest.png', affectez-la à la variable I, et convertissez-la en double précision.
2. Choisir comme germe le point  $[x \ y] = [198 \ 359]$

3. Appliquer l'algorithme de croissance de région, pour segmenter une région d'écart type 0.2, autour de ce germe-là. Vous obtiendrez une image (matrice) d'étiquettes  $J$ , qu'on appellera image segmentée.
4. Afficher l'image segmentée sur l'image initiale

```
figure, imshow(I+J);
```

5. Essayez à présent de segmenter une région dont vous sélectionnerez le germe au moyen de la souris.
6. Variez l'écart type 0.2, 0.3 ou 0.4. Commenter.
7. On voudrait à présent segmenter 4 régions, suivant les étapes suivantes :

- a) lire l'image 'medtest.png' et affecter-là à  $I$ , puis afficher dans la figure (3)
- b) initialiser la matrice des étiquettes à 0:  **$J_s = \text{zeros}(\text{size}(I))$**  ;
- c) effectuer une boucle pour  $i=1:4$  (4 est le nombre des régions)
  1. afficher la figure (3)
  2. sélectionner un germe avec la souris
  3. segmenter la région  $J$  autour du germe pour un écart type 0.2 (ou 0.3 ou 0.4 ou ...)
  4. Additionner la région segmentée à la matrice des étiquettes, après l'avoir étiquetée (à  $i$ , par exemple) :  **$J_s = J_s + J * i$**  ;
  5. convertir la matrice des étiquettes en une image couleur, puis afficher dans la figure(4)
  6. effectuer une pause avant de passer à la prochaine itération : **pause** ;

- Commenter les résultats.
- Est-ce que l'algorithme converge toujours?

### 3. Segmentation par la méthode des $k$ -moyennes ( $k$ -means)

- Cette méthode vise à diviser l'image en  $k$  classes (régions) statistiquement homogènes, dont les valeurs de niveaux de gris sont similaires au (proches du) centre (moyenne) de la classe.
- Cette similarité est estimée par calcul de distance entre la valeur du niveau de gris et du centre
- Il faut connaître à priori le nombre de classe (régions)

#### Commandes Matlab:

Permet de classer les points du **vecteur**  $X$  en  $k$  classes

- $\text{Idx}$  : résultat de la classification, vecteur des étiquettes (label vector)
  - $\text{Idx}$  est de la même taille que  $X$ .
  - L'étiquette (classification) de chaque point de  $X$  d'indice  $i$ , est contenue dans  $\text{Idx}$  au même indice  $i$ .
  - Les valeurs possibles de  $\text{Idx}$  sont les valeurs des étiquettes (classes) : 1, 2, 3, ...,  $k$ .
- $C$  : un vecteur de taille  $(1 \times k)$ , il contient les  $k$ -centres des classes.

```
[idx,C] = kmeans(X,k);
```

Réarrange les éléments de la matrice  $Z$  en une colonne  $X$ .

```
X=im2col(Z, size(Z));
```

- Avec cette syntaxe, si  $Z$  a une taille  $[m,n]$ , le vecteur  $X$  aura une taille  $m \times n$

---

`Z=col2im(X,[1 1],[m n]);` Réarrange les éléments du vecteur X en une matrice Z de taille [m,n].

---

### Manipulation:

1. Lire et afficher l'image à niveau de gris '`medtest.png`', affecter-la à la variable `I`
2. Réarranger les pixels de `I` en une seule colonne `II`
3. appliquer l'algorithme des k-means pour classer les points de `II`, en 2 régions. Vous obtiendrez le vecteur des étiquettes (`idx`) et les centres des classes (`C`)
  - quelles sont les centres des régions.
4. Réarranger le vecteur des étiquettes `idx`, en une matrice `Is`, puis afficher-la.
  - Observer les 2 régions
5. changer le nombre de classes (régions) à 3 puis à 4.
  - commenter.

### 4. Segmentation par la méthode de diffusion/fusion (split and merging)

- C'est une méthode qui vise d'abord à diviser l'image en régions homogènes selon un certain critère (prédicat)
- Puis opère à fusionner les régions obtenues et qui sont semblables selon ce même prédicat, en une seule région

### Commandes Matlab:

```
J = splitmerge(I, a,  
@predicate);
```

Permet de segmenter l'image `I` par la méthode de division/fusion.

- `a`: est la taille minimal des blocs d'image lors de l'opération de division. C'est un nombre 2 puissance : 1, 2, 4, 8, 16, ...
- `@predicate` : est une fonction qui contient les critères de division/fusion
- la matrice obtenue `J` est une matrice d'étiquettes.

### Fonctions Matlab:

```
function flag = predicate(region)  
    sd = std2(region);  
    m = mean2(region);  
    ecart_type = 20; %%%  
    val_min = 0; %%%  
    val_max = 200; %%%  
    flag = (sd > ecart_type) ...  
    & (m > val_min) & (m < val_max);
```

Aller dans la fonction "predicate" et changer les paramètres suivant par les valeurs voulues :

- `ecart_type` : écart type de la région à segmenter
- `val_min` : valeur minimale du niveau de gris dans la région à segmenter
- `val_max` : valeur maximale du niveau de gris dans la région à segmenter

## Manipulation:

---

1. Lire et afficher l'image à niveau de gris '`medtest.png`', affecter-la à la variable `I`
2. Appliquer l'algorithme de division/fusion,
  - en divisant l'image en des blocs, de taille minimale  $2^2$
  - selon les prédicats :
    - écart type = 20
    - valeur de gris minimale = 0
    - valeur de gris maximale = 200
  - Afficher le résultat de la segmentation
  - commenter
3. Changer la taille minimale de division :  $2^3$ ,  $2^4$ ,  $2^0$ ,  $2^5$ 
  - commenter
4. Pour une taille minimale de division =  $2^2$ , changer les valeurs des prédicats, et commenter.

## seuillage d'histogramme

```
I = imread('medtest.png');
imshow(I); figure, imhist(I);
thres = multithresh(I,4);
Is = imquantize(I, thres);
Irgb = label2rgb(Is);
figure;
imshow(Irgb)
```

## k-means

```
I = imread('medtest.png');
II=im2col(I,size(I));
[idx,C] = kmeans(II,4);
Is=col2im(idx,[1, 1],size(I));
Is=label2rgb(Is);
```

## 'growing\_region'

```
% cas1
I = im2double(imread('medtest.png'));
x=198; y=359;
J = regiongrowing(I,x,y,0.2);
figure, imshow(I+J);

% cas2
[x, y] = getpts; x=uint8(x); y=uint8(y);
J1 = regiongrowing(I,x,y,0.2);
figure, imshow(I+J+J1);

% cas 3
figure(3), imshow(I);
Js=zeros(size(I)); %image segmentée initiale à 0
for i=1:4
    figure(3)
    [x, y] = getpts; x=uint8(x); y=uint8(y);
    sig=0.4; %écart type = 0.2; 0.3; 0.4, ... ;
    J = regiongrowing(I,x,y,sig);
    Js=Js+J*i; %étiqueter la région segmentée J avec i
    Js_rgb=label2rgb(Js);
    figure(4), imshow(Js_rgb));
    pause,
end
```

## Division/fusion

```
I = (imread('medtest.png'));
J = splitmerge(I, 2^2, @predicate);
Jrgb=label2rgb(J);
figure, imshow(Jrgb);

function flag = predicate(region)
    sd = std2(region);
    m = mean2(region);
    ecart_type = 20;
    val_min = 0;
    val_max = 200;
    flag = (sd > ecart_type) & (m > val_min) & (m < val_max);
```