Département ATE
Master 1 en Systèmes des Télécommunications
Semestre 2 / Mai 2020
Module : Traitement d'Image

TP 4
Détection des Contours

But

Aborder la détection des contours dans une image, au travers :
- Les opérateurs locaux approximant le gradient : Prewitt, Sobel et Roberts
- Les opérateurs locaux approximant le Laplacien : Laplacien et LoG
- Filtre optimal (global) de Canny

1. *Détection des Contours au moyen d'Opérateurs Locaux Approximant le Gradient*

- Pour détecter les contours on abordera en premiers lieu les opérateurs locaux approximant le Gradient à savoir, les opérateurs de :

  - Prewitt - Sobel - Roberts

- On va créer chacun des opérateurs, puis on verra :
  - Comment le convoluer avec l'image pour calculer le gradient
  - puis comment extraire les contours par seuillage du module du gradient

Les commandes Matlab

`dx=fspecial('prewitt');`
 	
`dx=fspecial('sobel');`
	– Permet de créer un des noyaux de l'opérateur de Prewitt (ou Sobel). Celui qui approxime le Gradient dx dans le sens des x.
	– Le 2ème noyau, approximant le gradient dans le sens des y, s'obtient en calculant le transposé de dx : dy=dx'.

`Iedge = edge(I, operator);`	– Permet d'extraire les contours de l'image I via l'opéarteur indiqué dans « opertor », à savoir : 'Prewitt', 'Sobel' ou 'roberts'.
`Iedge = edge(I, operator, thres);`	– On peut rajouter un paramètre de seuillage thres
`[Iedge, thres]=edge(I, operator);`	– Comme on peut extraire le paramètre de seuillage optimal calculé par la commande edge Dans tous les cas, l'image contour obtenue est affectée à Iedge

`Thres=graythres(I)`	`Permet de calculer un seuil optimal de seuillage, à partir de l'histogramme en utilisant la méthode d'Otsu.`

Manipulation

1. Lire et afficher l'image à niveau de gris **`'circuit.tif'`**, affecter-la à la variable **`I`**

– convertissez-là en simple ou double précision : **`I=im2single(I);`** ou **`I=im2double(I);`**

2. **Méthode 1** :

a) Calculer le noyau de l'opérateur de « **`prewitt`** », **`dx,`** approximant le gradient dan les sens des x.

b) Calculer le noyau **`dy,`** approximant le gradient dans le sens des y

– Afficher **`dx et dy,`** et vérifier avec les noyaux vus en cours.

c) **Convoluer** l'image **`I`** avec les noyaux **`dx`** et **`dy`** , pour avoir son gradient

$$\nabla I = \begin{cases} dxI = I * dx \\ dyI = I * dy \end{cases}.$$

Revenir au TP4 pour revoir la convolution sur une image.

d) Calculer le module du gradient **`∇I, et`** affecter-le à la variable **`mod_gradI`**

$$\texttt{mod\_gradI} = |\nabla I| = \sqrt{dxI^2 + dyI^2}$$

e) Afficher **`dxI, dyI et |∇I|`** `dans la même figure.`

– inverser la palette des couleurs gris en utilisant la commande : **`colormap(1-gray),`**

– Commenter les images obtenues

f) Pour obtenir **l'image contour**, qui est une **image binaire**, on effectuera **un seuillage sur `|∇I|`.**

- Les valeurs élevées de **`|∇I|`** et supérieures à un seuil, correspondent aux points contours,
- Les valeurs faibles de **`|∇I|`**, ne correspondent pas aux points contours,

$$\begin{cases} |\nabla I|(i,j) \geq seuil & I_{edge}(i,j) = 1 \\ |\nabla I|(i,j) < seuil & I_{edge}(i,j) = 0 \end{cases}$$

– Prendre comme seuil la valeur **`0.35`**.

– Pour réaliser ce seuillage, écrire les lignes commandes suivantes :

```
Iedge=zeros(size(mod_gradI));
Iedge(mod_gradI>=0.35)=1;
```

– Afficher l'image contour dans la même figure que l'image originale. Commenter.

– Varier le seuil et observer les résultats puis commenter.

– Calculer un seuil optimal au moyen de la commande **`"graythresh".`** Utiliser-le dans le seuillage. Afficher et commenter.

g) Manier à présent d'autres filtres : **`Sobel`** et **`Roberts`**.

– pour Roberts, il faudra introduire soit même les opérateurs : $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ et $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

3. Méthode 2 :

On utilisera la commande « **`edge`** », qui implicitement :

- Calcule les noyaux de l'opérateur
- Calcule le gradient de l'image
- Effectue le seuillage.

a) Calculer l'image contour $\mathbf{I}_{\mathbf{edge}}$ de l'image **`I`** au moyen de la commande « **`edge`** », et via respectivement les opérateurs, **`Prewitt, Sobel`** et **`Roberts.`** (*)
   - Utiliser pour le seuillage la valeur **`thresh=0.06.`**
   - Afficher les résultats et comparer entre les détections obtenus par chacun des 3 opérateurs..

b) Varier le seuil, **`thresh=0.08, 0.07, 0.06, 0.05, 0.04, 0.03.`**
   - **C**ommenter.
   - Parmi ces valeurs de seuil, et pour chaque opérateur, quelle est valeur qui donne une détection optimale

c) Utiliser la commande « **`edge`** », sans introduire une valeur du seuil, et extrayez la valeur optimale calculée par cette commande.
   - Comparer entre les valeurs seuils déduites dans la question b).

2. *Détection des Contours au moyen d'Opérateurs Locaux Approximant le Laplacian*

- A présent, on abordera les opérateurs locaux approximant le Laplacian à savoir, les opérateurs de :

  - Laplacian - LoG (Laplacian Of Gaussian)

- On va créer chacun des deux opérateurs,
  - qu'on convoluera avec l'image pour calculer le Laplacian
  - puis on extraira les points contours par opération de seuillage

(*) *: Avec la commande "edge", vous pouvez laisser l'image I en codage 8bits, sans la convertir en double ou simple précision*

Les Commandes Matlab :

`h=fspecial('Laplacian');`	Permet de créer le noyau de l'opérateur de Laplacian
`h=fspecial('LoG',n,sigma);`	Permet de créer le noyau de l'opérateur de LoG, de taille n et pour une gaussienne de variance sigma. (par défaut n=5 et sigma=0.5)
`Iedge = edge(I, 'LoG', thres );`	– Permet d'extraire les contours de l'image I via l'opérateur « LoG » – On peut mettre ou omettre la variance sigma de la fonction gaussienne. Si on l'omet, la valeur par défaut est sigma=2.
`Iedge=edge(I,'LoG',thres, sigma);`	– On peut rajouter un paramètre de seuillage thres
`[Iedge,thres]=edge(I,'LoG',sigma);`	– Comme on peut extraire le paramètre de seuillage optimal calculé par la commande edge Dans tous les cas, l'image contour obtenue est affectée à Iedge

Manipulation

1. **Opérateur Lapalcian**:
   a) Calculer le noyau de l'opérateur « **`Laplacian`** », **`h`**, approximant le Lapalcien. Afficher et vérifier avec le noyau vu en cours.
   b) Convoluer l'image **`I`** avec le noyau **`h`** , pour avoir son Laplacien Δ**`I = I*h.`**
   c) Pour obtenir **l'image contour**, on cherchera les points où il y'a passage par zéro, autrement dit il y'a un changement de signe entre les valeurs des points adjacents.

$$\begin{cases} \Delta I(i,j) - \Delta I(i+1,j) \leq seuil & I_{edge}(i,j) = 1 \\ \Delta I(i,j) - \Delta I(i,j+1) \leq seuil & I_{edge}(i,j) = 1 \\ \Delta I(i,j) - \Delta I(i+1,j+1) \leq seuil & I_{edge}(i,j) = 1 \\ sinon & I_{edge}(i,j) = 0 \end{cases}$$

   – Ce changement de signe on le cherchera pas au niveau du zéro (sinon il y'aura sur-detection), on comparera de ce fait à une autre valeur différente de zéro ($seuil \neq 0$).
   Prenez comme $seuil = 4\%\ du\ maximum\ de\ |\Delta I|$

```
s=max(abs(ΔI(:)))/25;
Iedge=zeros(size(I));
Iedge(abs(A)<s & abs(B)<s & abs(C)<s)=1;
```

   – Afficher l'image contour et commenter.

2. Opérateur LoG:

On utilisera la commande « **edge** », qui implicitement :

- Calcule le noyau de l'opérateur
- Calcule le Laplacian de l'image
- Effectue le seuillage.

Cette commande n'est valable que pour l'opérateur `LoG`*, et ne l'est pas pour le* `Laplacien`*.*

a) Calculer l'image contour $\mathbf{I}_{\mathbf{edge}}$ de l'image **I** au moyen de la commande « **edge** » via l'opérateur **LoG**
   - Utiliser pour le seuillage la valeur **thresh=0.003,** ne mettez rien pour sigma.

b) Varier le seuil, **thresh=0.001, 0.003, 0.005.** Afficher et commenter.

c) Utiliser la commande « **edge** », sans introduire une valeur du seuil, et extrayez la valeur optimale calculée par cette commande.

d) La valeur par défaut de la variance de la gaussienne du **LoG** est **sigma=2.** Variez-là par les valeurs : **1, 1.5, 2, 2.5, 3, 3.5.** Afficher et Commenter.

e) Au vue des résultats vus dans les deux sections 1. et 2., confirmer la préférence de l'opérateur LoG au Lapalcian.

Détection des Contours au moyen du Filtre Optimal de Canny

- A présent, on abordera une méthode globale de détection de contours : le filtre optimal de Canny.
  - Cette méthode approxime le Gradient.
  - Le seuillage utilisé est par hystérésis, de ce fait, deux seuils sont nécessaires, un, pour les valeurs minimales, et un autre, pour les valeurs maximales.

- Sous Matlab, on utilisera la commande :

`Iedge = edge(I, 'canny');`	– Permet d'extraire les contours de l'image `I` via le filtre de « `canny` »
`Iedge=edge(I,'canny',thres);`	– On peut rajouter les deux paramètres du seuillage par hystérésis `thres=[th_min, thr_max]`
`[Iedge, thres]=edge(I, 'canny');`	– Comme on peut extraire les paramètres optimaux de seuillage par hystérésis calculés par la commande `edge` L'image contour obtenue est affectée à `Iedge`

Manipulation

1. Calculer l'image contour $\mathbf{I}_{\mathbf{edge}}$ de l'image **I** au moyen de la commande « **edge** »
   - Utiliser pour le seuillage la valeur **thresh=[0.1, 0.3].**
2. Varier le seuil, **thresh=[0.1, 0.3], [0.1, 0.4], [0.1, 0.5], [0.05, 0.3], [0.15, 0.3], [0.2, 0.3].** Afficher et commenter.
3. Utiliser la commande « **edge** », sans introduire de valeurs pour les seuils, et extrayez les valeurs optimales calculée par cette commande.

Synthèse

1. Lire l'image `'cameraman.tif'` (laisser là en codage 8bits)
2. Bruiter l'image par un bruit gaussien de variance **0.005**.
3. Appliquer les 3 opérateurs (Prewitt, Sobel, LoG) ainsi que le filtre de Canny (Utiliser le calcul automatique des seuils optimaux)
   - Afficher et comparer.
4. Lisser l'image bruitée avec un filtre gaussien (taille et variance au choix).
5. Appliquer à nouveau les 4 détecteurs de contours de la question 3.
   - Afficher et comparer.

edge detecion (gradient) : méthode 1

```
I=imread('circuit.tif');
figure(1);
subplot(121); imshow(I); title('image originale');
I=im2single(I);
dx=fspecial('prewitt');% ou bien taper directement [1 1 1;0 0 0;-1-1 -1]
dy=dx';
dxI=conv2(I,dx,'same');
dyI=conv2(I,dy,'same');
mod_gradI=sqrt((dxI).^2+double(dyI).^2);
%
figure(2);
subplot(131), imshow(abs(dxI)); title('dxI');
subplot(132),  imshow(abs(dyI)); title('dyI');
subplot(133),  imshow((mod_gradI)); title('gradient de I');
colormap(1-gray), % pour avoir les valeurs max en noir sur fond blanc
%
Iedge=zeros(size(mod_gradI));
Iedge(mod_gradI>=0.35)=1;
figure(1),
subplot(122), imshow(Iedge); title('image contour');
```

edge detecion (gradient) : méthode 2

```
thresh=0.06;  %0.08, 0.07, 0.06, 0.05, 0.04, 0.03, % varier le seuil
Iedge1=edge(I,'prewitt', thresh);
Iedge2=edge(I,'sobel',thresh);
Iedge3=edge(I,'roberts',thresh);
figure(3),
subplot(131), imshow(Iedge1);
subplot(132), imshow(Iedge2);
subplot(133), imshow(Iedge3);
%
[Iedge1, thresh1]=edge(I,'prewitt'); thresh1
[Iedge2, thresh2]=edge(I,'sobel'); thresh2
[Iedge3, thresh3]=edge(I,'roberts'); thresh3
figure(4),
subplot(131), imshow(Iedge1);title('prewitt');
subplot(132), imshow(Iedge2);title('sobel');
subplot(133), imshow(Iedge3);title('roberts');
```

edge detecion (Laplacian)

```
h=fspecial('laplacian');
LapI=conv2(I,h,'same');
figure(5)
subplot(131), imshow(I);
subplot(132), imshow(mat2gray((LapI/max(max(abs(LapI))))));
%
LapIi=LapI([end,1:end-1],:);
LapIj=LapI(:,[end,1:end-1]);
LapIij=LapIi(:,[end,1:end-1]);
A=LapI-LapIi;
B=LapI-LapIj;
C=LapI-LapIij;
```

```
kf=find(abs(A)<0.03 & abs(B)<0.03 & abs(C)<0.03 );
Iedge=zeros(size(I));
Iedge(kf)=1;
%
subplot(133), imshow(Iedge);
```

edge detecion (LoG)

```
thresh=0.003; %0.001, 0.003, 0.005
Iedge=edge(I,'log',thresh);
% [Iedge, thresh]=edge(I,'log'); thresh
% sigma=2; Iedge=edge(I,'log',[],sigma); %1, 1.5, 2, 2.5, 3, 3.5
Figure(6),
imshow(Iedge);
```

edge detecion (Canny)

```
thresh=[0.1, 0.3];
Iedge=edge(I,'canny',thresh);
%[Iedge, thresh]=edge(I,'canny'); thresh
Figure(7),
imshow(Iedge);
```

Comparaison

```
% I=imnoise(I,'gaussian',0,0.005); % bruitage
% h=fspecial('gaussian'); I=imfilter(I,h); % lissage
figure(8), imshow(I);
% détection avant et après lissage
[Iedge1, thresh1]=edge(I,'prewitt'); thresh1
[Iedge2, thresh2]=edge(I,'sobel'); thresh2
[Iedge3, thresh3]=edge(I,'Log'); thresh3
[Iedge4, thresh4]=edge(I,'canny'); thresh4
figure(9),
subplot(121), imshow(Iedge1);title('prewitt');
subplot(122), imshow(Iedge2);title('sobel');
figure(10),
subplot(121), imshow(Iedge3);title('LoG');
subplot(122), imshow(Iedge3);title('Canny');
```