AIIP
(mai.nguyen-verger@cyu.fr)

Filtrage et convolution pour la détection des caractéristiques

Rappel sur l'approche de dérivation pour la détection de contours

1) Approche **gradient** : détermination des extréma locaux dans la direction du gradient.

2) Approche **laplacien** : détermination des passages par zéro du laplacien (ou du changement des signes de deux pixels adjacents en y, x).

3) Détection de contours à l'aide de convolution avec les **filtres dérivateurs** Prewitt et Sobel dont les noyaux sont les suivants :

   Soit *hx* et *hy* : masques vertical et horizontal de taille 3x3 pixels,

   Filtre de Prewitt :

$$hx = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad hy = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

   Filtre de Sobel :

$$hx = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad hy = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

4) Rappel sur l'**algorithme** de détection de contours à l'aide de gradients :

   1. Appliquer un filtre dérivateur sur une image originale afin de calculer les gradients de l'image en *x* et en *y*, nommés *Ix* et *Iy* respectivement (qui sont en lien avec les contours verticaux et horizontaux respectivement).

   2. Afficher les gradients *Ix* et *Iy* séparément.

   3. Calculer la norme du gradient : $Ig = (\textbf{\textit{sqrt}}\ (Ix^2 + Iy^2\ ))$ et afficher l'image de *Ig*.

   4. Détecter des contours de l'image originale en appliquant le seuillage sur la norme du gradient : $Ci = (Ig \geq Seuil)$, et afficher l'image de contours *Ci* (qui est une image binaire : $Ci(y,x)=1$ indique que le pixel (y,x) se trouve sur le contour, $Ci(y,x)=0$ sinon).

   (la valeur de la variable « *Seuil* » est choisi d'une manière empirique afin d'obtenir les meilleurs contours. Suggestion : chercher les valeurs maximale et minimale de la norme du gradient *Ig* par ***max***(***max***(*Ig*)) et ***min***(***min***(*Ig*)) afin de choisir le seuil optimal entre ces deux valeurs).

2

5) Rappel sur le **Produit de convolution discret 2D** :

$$C(y,x) = \sum_{k} \sum_{l} I(y-k, x-l) h(k,l)$$

La convolution affecte à chaque pixel (y,x) un niveau de gris **C(y,x)** estimé par la **somme** des niveaux de gris des **pixels de son voisinage pondérés** par des **coefficients du filtre**

Rappel sur Matlab**:**

1) Les niveaux de gris d'images sont entre [0,255] ou entre [0,1] avec noir (0) et blanc (255 ou 1).

2) Les opérations dans les images sont réalisées souvent avec des données de type « double » (exemple : log). S'assurer que les opérations s'effectuent sur une image I de ce type (en utilisant ***double(I)*** par exemple).

3) Changement de type des valeurs d'une image :
uint8( ), uint16( ) : entiers non signés 8, 16 bits
double( ) : double précision
logical( ) : binaire.

4) Tester trois fonctions d'affichage : **imshow(), image(), imagesc() :**
imshow() affiche I de type d'entier, si I de type double il faut **imshow**(**uint8**(I))

5) Les fonctions utiles: ***abs*(), *size*(), *sum(sum())*, *zeros*()**, opération matricielle terme par terme: .* (point étoile pour la multiplication terme par terme), **./** (point barre oblique).

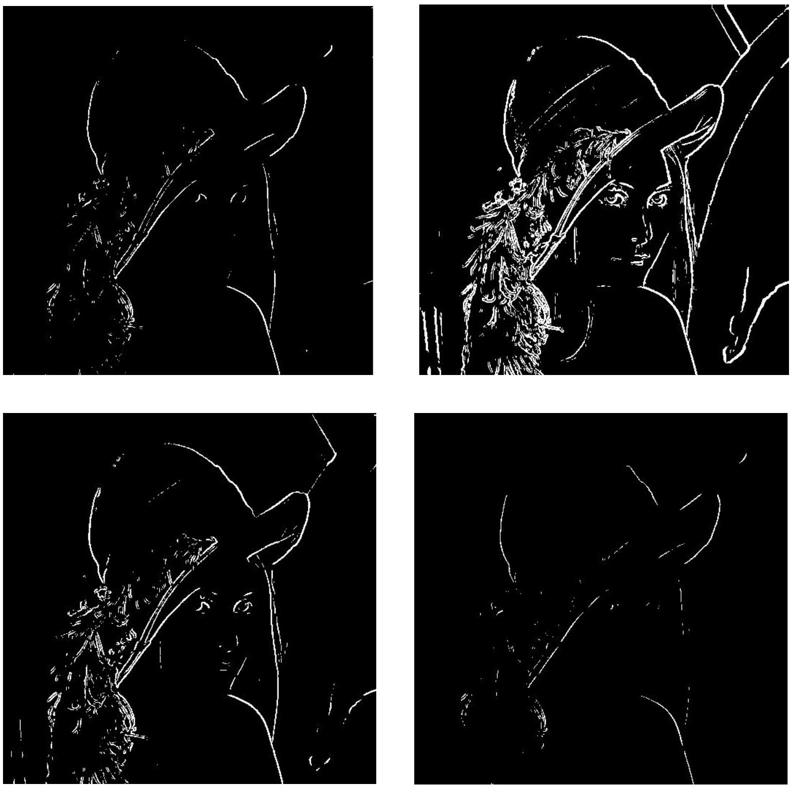
6) **Attention à l'effet de bord : on peut ajouter les zéros autour de l'image avant d'effectuer les opérations.**

 3

Partie 4. Détection de contours

1) Détection de contours des triangles du AIIP_1, Partie 1(a) par le filtre Prewitt.
2)
   2.1) Détection de contours de l'image dans le fichier « lena.jpg » par le filtre Sobel.
   2.2) Varier les seuils, puis observer et expliquer les résultats obtenus avec différents seuils : seuil **fort** et seuil **faible**
   Comment obtenir les contours plus ou moins fins comme dans la figure ci-après.
   2.3) Détection de contours par le seuillage **hystérésis (**algorithme et programme)

Image originale



Différentes images de contours de l'image originale « lena »

3) Détection de contours de l'image « lena » par les filtres Laplaciens suivants :

$$\boldsymbol{h_{L1}} = \begin{pmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{-4} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix}, \quad \boldsymbol{h_{L2}} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{-8} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

4

Partie 5. Filtrage optimal pour la détection des contours

I) L'inconvénient de l'approche de dérivation est qu'elle est très sensible aux bruits (un point bruité isolé peut devenir un point de contour).

Un autre type d'approche repose sur la définition de critères d'optimalité de la détection de contours. Ces critères débouchent sur des **filtres optimaux**.

II) Les **critères d'optimalité** sont :

- Bonne détection : le contour doit être détecté, il faut minimiser les fausses détections.
- Bonne location : le contour doit être localisé avec précision, il s'agit de minimiser la distance entre les points détectés et le vrai contour.
- Unicité de réponse : il s'agit de minimiser le nombre de réponse pour un seul contour.

Exemples des méthodes de cette approche :

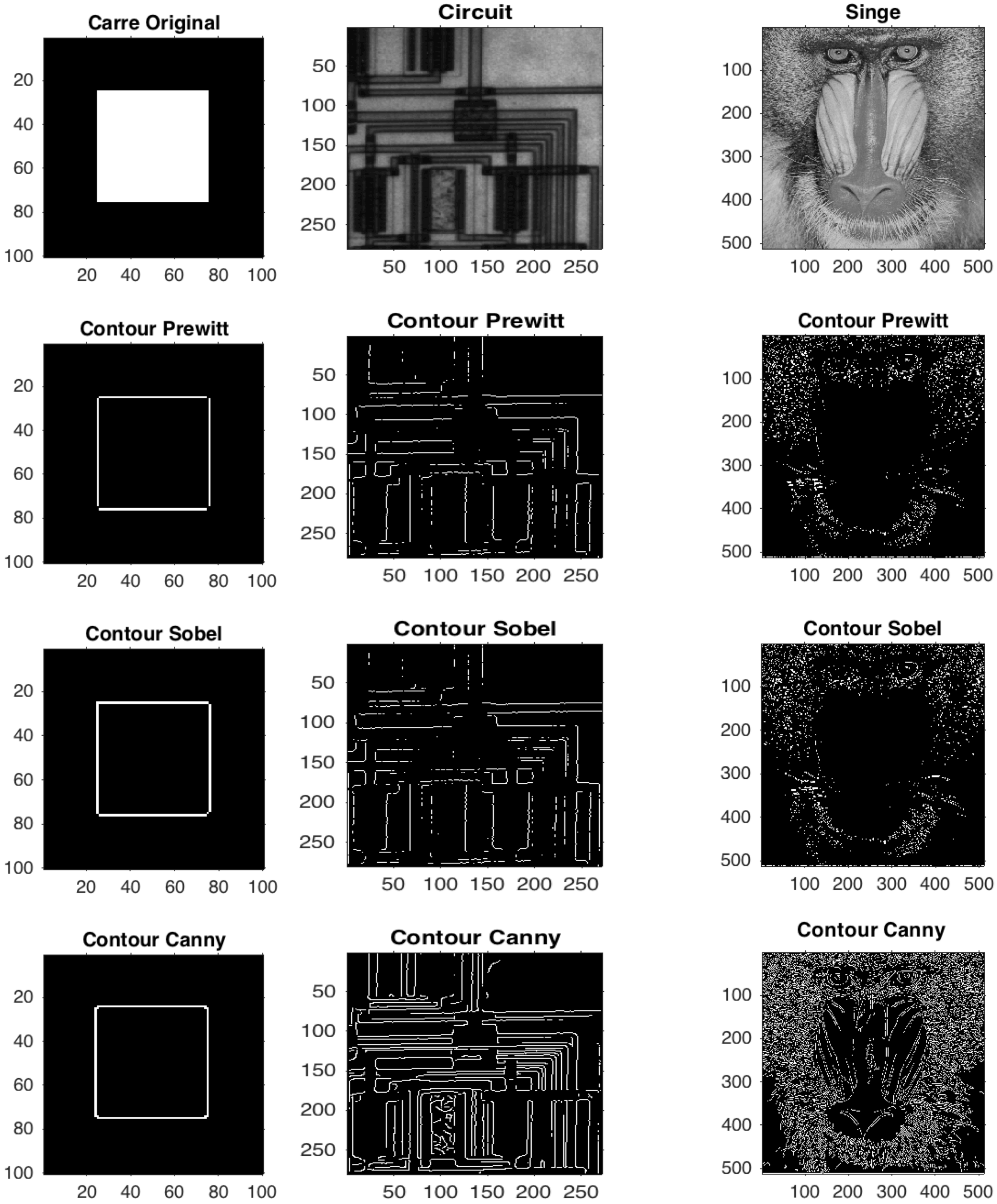
1. Filtre de Canny
2. Filtre Gaussien, Différence de Gaussiens (DOG), Lalacien de Gaussien (LOG).
3. Filtre de Shen-Castan
4. Filtre de Deriche

III) Travail demandé :

1) Choisir un des filtres optimaux.
2) Écrire les formules mathématiques de ce filtre.
3) Décrire les propriétés de ce filtre.
4) Préciser le réglage des paramètres dans ces formules.
5) Tracer l'allure de ce filtre.
6) Programmer ce filtre sur une/des images réelles où les filtres dérivateurs simples échouent (image d'intensité progressive, empreint digitale, 'mandrill.png', etc.).

(La recherche bibliographique sur le Web ou à la bibliothèque est nécessaire pour ce travail).

5


Carre Original
Circuit
Singe
Contour Prewitt
Contour Prewitt
Contour Prewitt
Contour Sobel
Contour Sobel
Contour Sobel
Contour Canny
Contour Canny
Contour Canny

 6

Partie 6. Détection des points d’intérêt

Écrire un programme de détection des points d’intérêt par le détecteur de Harris (sans puis avec convolution par un filtre Gaussien) et l’appliquer sur les images de test qui sont des triangles du AIIP.1, puis ‘cameraman.tif’.

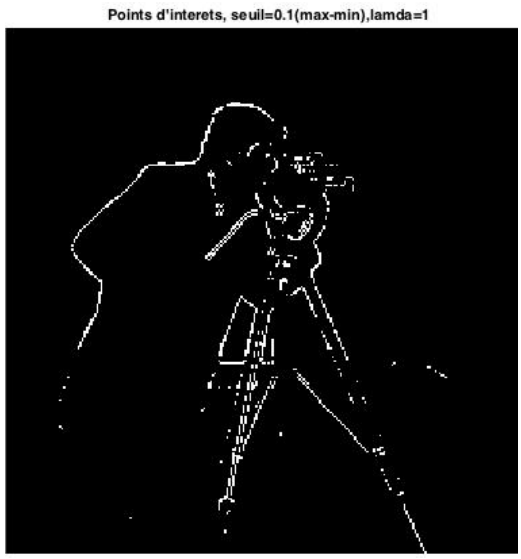
Rappel : formule d’Harris :

$$\boldsymbol{R} = \langle \boldsymbol{Ix^2} \rangle \langle \boldsymbol{Iy^2} \rangle - \langle \boldsymbol{IxIy} \rangle - \boldsymbol{\lambda}(\langle \boldsymbol{Ix^2} \rangle + \langle \boldsymbol{Iy^2} \rangle)^2$$

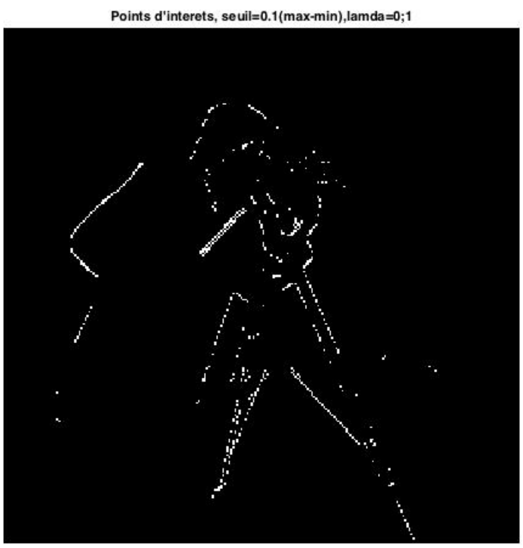
où *Ix* (respectivement *Iy*) est la dérivée première de *I(y,x)* en *x* (respectivement en *y*) ;
<F> représente la convolution de F par un filtre Gaussien, $\boldsymbol{\lambda}$ est une constante.

R étant grand sur les coins, la détection des points d’intérêt est donc faite par la recherche des maxima locaux de ***R***.

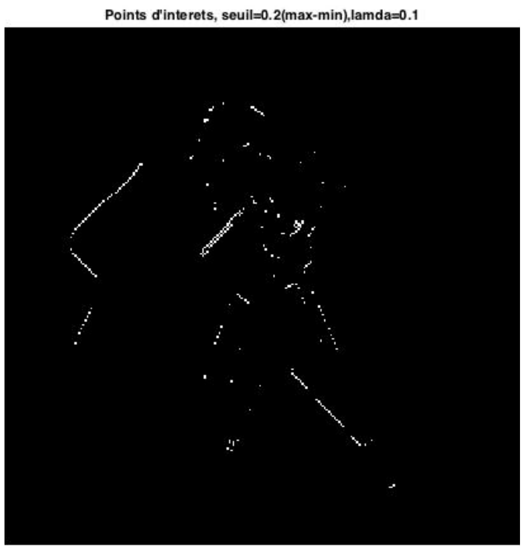
L’avantage de cette méthode est que seules dérivées premières sont calculées.



Seuil= 0.1(max) ; lamda=1



Seuil= 0.1(max) ; lamda=0.1



Seuil= 0.2(max) ; lamda=0.1