Traitement d'image

TP N° 3 : Détection de contour

1. Introduction

Ce TP a pour objet l'évaluation les performances de deux détecteurs de contour : l'algorithme de Canny et la méthode basée sur le passage par zéro du laplacien de l'image. L'image test utilisée sera le photophore des tps précédent.

La détection de contour par la méthode de Canny sera réalisée à l'aide d'une fonction Matlab fournie :

```
>> edg = canny(im,seuil) ;
```

La variable **im** correspond à l'image source sur laquelle le contour est détecté, la variable **seuil** est le seuil haut de l'algorithme de Canny.

On rappelle que la méthode de détection de contour par la méthode du passage par zéro du laplacien est basé sur quatre étapes principales :

1. Calcul du laplacien de l'image **imlap** par convolution avec un masque laplacien donné par :

0	1	0
1	-4	1
0	1	0

2. Création d'une image de polarité **impol** en appliquant la règle suivante :

impol = 0 si **implap** > 0 , **impol = 1** si **implap** ≤ 0

3. Création d'une image **imzero** des passages par zéro :

imzero(i,j) = 1 si **(i,j)** correspond à une transition (0 1) ou (1 0) de **impol** (0 sinon)

4. Le résultat est ensuite affiné en éliminant les points de passage par zéro associés à un gradient faible.

2. Manipulation

2.1. Réalisation du détecteur de contour

Dans cette première partie il est demandé de réaliser une fonction matlab permettant de calculer l'image des contours par la méthode du passage par zéro du laplacien. Une fonction matlab est un fichier texte (script interprété) qui débute par une instuction précisant le nom de la function est les paramètres passés à la fonction. Dans le cas du détecteur de contour demandé, les paramètres de passage seront l'image source et le seuil de l'étape 4, ce qui donne :

```
function result = ContourLaplacien(ImSrc, seuil)
```

Le reste du programme est constitué des instructions matlab permettant de réaliser le traitement. Le résultat final doit être sauvegarder dans la variable **result** afin qu'il puisse être récupéré lors de l'appel. Le fichier sera sauver avec le même nom que la fonction et l'extension .m c'est à dire ContourLaplacien.m.

Indications.

Afin de réaliser les étapes 2., 3. et 4. de l'algorithme, il est notamment possible d'utiliser avec matlab des opérations booléennes qui seront appliquées à chaque pixel.

Exemple :

```
>>pos = im > 20 ;
```

permet de calculer une image pos pour laquelle les pixels sont des valeurs booléennes (0 ou 1) correspondant à la règle suivante :

pos(i,j) = 1 si **im(i,j) > 20, pos(i,j) = 0** sinon.

Pour réaliser l'étape 3 il est possible de comparer l'image **imzero** avec une version décalée d'elle même. Pour cela il est nécessaire d'extraire deux sous images de **imzero**. Cette opération est réalisée simplement avec matlab.

Exemple :

```
>>imzero1 = imzero(1:20,50:70) ;
```

2.2. Comparaison des détecteurs de contour

Dans cette partie nous allons comparer les deux détecteurs de contour par rapport à leur sensibilité au bruit. Le critère utilisé sera le taux de pixels mal classés.

Pour cela il est nécessaire de calculer au préalable une image de référence en appliquant les deux détecteurs sur l'image du photophore sans bruit :

```
>> refcanny = canny(im,seuil) ;
>> reflapla = ContourLaplacien(im,seuil) ;
```

Ces images de référence seront ensuite comparées aux résultats obtenus (respectivement pour chacun des détecteurs) avec différents bruits. Le taux d'erreur est déterminer en calculant le nombre de pixels mal détectés c'est à dire ne sommant les fausses détections et les absences de détection.

Exemple : pour un écart type sigma = 5, le tau d'erreur pour l'algorithme de Canny est calculé par :

```
>>tauxCanny05 = sum(sum(refcanny ~= canny05)) ;
```

N.B. :

- canny05 est ici l'image des contours calculée pour un bruit d'écart type 05.
- Le symbole ~= (tilde égal) est l'opérateur logique « différent de ».

Calculer les taux d'erreur pour les deux algorithmes pour des bruits d'écarts types respectifs de 05, 10, 15, 20. En déduire l'algorithme le plus sensible au bruit.