# Détection de contours

OBJECTIF : préparation du troisième exercice noté (semaine 11)

## Présentation

Cette activité doit permettre de mettre en œuvre la suite des traitements permettant d'extraire les contours d'une image monochrome par des méthode dérivatives du premier ordre :

1. le calcul des dérivées partielles suivant l'axe des $i$ ou l'axe des $j$ au moyen des filtres :
   — dérivateurs 1D,
   — filtres de Prewitt ou de Sobel,
   — filtre de Deriche ;
2. le calcul de la norme du gradient ;
3. le calcul des maximums locaux du gradient ;
4. le seuillage simple ou par hystérésis de la norme ou des maximums locaux du gradient.

Le dérivateurs 1D sont mis en œuvre au moyen de convolutions 1D appliquées horizontalement ou verticalement. Les filtres 2D (Prewitt, Sobel et Deriche) sont séparables et sont donc eux aussi mis en œuvre par des enchaînements de convolutions 1D.

Pour appliquer une convolution verticalement à une image $I$, on procèdera de la manière suivante :

— construire $I'$ le symétrique de l'image $I$ par rapport à son antidiagonale : $I'(i,j) = I(j,i)$ ;
— appliquer la convolution horizontalement sur $I'$ ;
— prendre à nouveau le symétrique du résultat obtenu : $I''(i,j) = I'(j,i)$.

Par conséquent, tous les calculs de dérivées partielles seront mis en œuvre à partir de **seulement deux opérations de base** :

— la convolution horizontale avec un noyau de largeur 1,
— la symétrisation d'une image par rapport à son antidiagonale.

On pourra utiliser la stratégie de son choix pour gérer les bords de l'image lors de la mise en uvre de la convolution.

Remarque. Les valeurs des pixels des images sont exprimées au moyen de caractères sans signe (`unsigned char` : valeurs comprises entre 0 et 255). Le résultat de l'application des filtres utilisés est un nombre réel de l'intervalle $[-255, 255]$. On convertira le canal de l'image à traiter en une matrice de flottants (`float`) avant traitement, puis on le reconvertira en caractères sans signe pour l'affichage une fois le ou les traitements terminés. Lors de cette seconde conversion, on utilisera deux stratégies possibles pour traiter les valeurs négatives :

— valeur absolue : si $x < 0$ alors $x \to -x$ ;
— normalisation : $\forall x \in [-255, 255]$ , $x \to \frac{1}{2}x + 127$.

# Description des activités

## 1 Application de convolution 1D et affichage du résultat

◦ Le module `convol1d` permet de mettre en œuvre les convolutions 1D avec de noyaux de largeur 1. Il contient les deux fonctions :

- `convol1d_apply_3` : application horizontalement de la convolution 1D de noyau $\begin{pmatrix} a & b & c \end{pmatrix}$ sur une image monochrome de taille $WIDTH \times HEIGHT$ ;
- `convol1d_swap_ij` : construction du symétrique d'une image par rapport à son antidiagonale.

◦ Le module `channel` contient les deux fonctions de conversion entier → réel et réel → entier des valeurs des pixels :

- `channel_to_float` : conversion en flottant de l'image entière ;
- `channel_to_uchar` : conversion en caractère sans signe de l'image flottante.

La seconde fonction possède un paramètre pouvant pendre les valeurs `Cdm_absval` ou `Cdm_normalize` et permettant de choisir la stratégie de conversion (valeur absolue ou normalisation). Ces deux fonctions retournent une zone mémoire allouée dynamiquement.

Activité **1**

*Écrire les deux fichiers `convol1d.c` et `channel.c` (les interfaces `convol1d.h` et `channel.h` sont fournies). Les tester avec le programme `test-1.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

### Conseils et remarques

- avant d'effectuer une conversion du type `float` vers le type `unsigned char`, faire par précaution une troncature entre 0 et 255, puis calculer l'arrondi entier au plus proche (fonction `lround`) ;
- attention à bien appliquer le symétrique du noyau de convolution ;
- attention, lors de la construction du symétrique d'une image, les indices de l'image résultante sont inversés, et donc également la largeur et la hauteur ;
- le programme `test-1` met en œuvre une dérivation partielle au moyen de l'opérateur | -1 | 1 | en appliquant le noyau | 0 | -1 | 1 | ;
- il est possible d'utiliser `test-1` avec l'option `-i` pour obtenir l'inversion de l'image résultante (contours noirs sur fond blanc).

## 2 Mise en œuvres des dérivées partielles

◦ Les filtres de Prewitt et de Sobel (voir diapositive 15 du thème B-05) calculent des dérivées partielles d'une image $I$ en itérant l'application de deux convolutions 1D :

— un lissage dans la direction orthogonale à la dérivation ;

— une dérivation dans la direction de la dérivation.

Les noyaux des lissages sont respectivement [1] [1] [1] et [1] [2] [1]. Le noyau de dérivation est le même pour les deux filtres : [-1] [0] [1].

Le module `pderiv` contient plusieurs fonctions de calcul de dérivées partielles discrètes d'une image :

— `pderiv_d2_di` et `pderiv_d2_dj` : calcul des dérivées partielles par rapport à $i$ et par rapport à $j$ au moyen du noyau [-1] [1] ;

— `pderiv_d3_di` et `pderiv_d3_dj` : idem avec le noyau [-1] [0] [1] ;

— `pderiv_prewitt_di` et `pderiv_prewitt_dj` : idem avec le filtre de Prewitt ;

— `pderiv_sobel_di` et `pderiv_sobel_dj` : idem avec le filtre de Sobel.

Activité **2**

*Écrire le fichier `pderiv.c` (l'interface `pderiv.h` est fournie). Le tester avec le programme `test-2.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

### Conseils et remarques

— pour écrire les codes des deux premières fonctions, il suffit de reprendre la mise en œuvre de ces deux dérivations dans le fichier `test-1.c` ;

— les autres fonctions sont mises en œuvre sur le même principe.

◦ Le filtre de Deriche (voir diapositive 18) est construit de la même manière mais avec des convolutions mises en œuvre de manière récursive (formules en annexe du diaporama, diapositives 41 et 42). L'implémentation de ces deux convolutions est fournie dans le fichier `deriche.c` :

— `deriche_apply_d` : application du dérivateur horizontal ;

— `deriche_apply_s` : application du lissage horizontal.

Activité **3**

*Ajouter au fichier `pderiv.c` les fonctions `pderiv_deriche_di` et `pderiv_deriche_dj`. Le tester avec le fichier `test-3.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

## 3 Calcul de la norme et des maximums locaux du gradient

◦ La norme euclidienne du gradient d'une image $I$ se calcule à partir de ses deux dérivées partielles :

$$||\nabla I|| = \sqrt{\frac{\partial I}{\partial i}^2 + \frac{\partial I}{\partial j}^2}$$

Activité **4**

*Écrire le fichier `gradient.c` qui contient la fonction `gradient_norm_eucl` calculant la norme euclidienne du gradient d'une image monochrome (l'interface `gradient.h` est fournie). Le tester avec le programme `test-4.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

◦ Pour affiner les contours, on calcule les maximums locaux de la norme du gradient dans la direction du vecteur gradient. La méthode est décrite sur la diapositive 31.

Activité **5**

*Ajouter au fichier `gradient.c` la fonction `gradient_local_max` calculant les maximums locaux selon cette méthode La tester avec le fichier `test-5.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

## 4 Seuillage simple et par hystérésis

◦ On termine en général la construction de l'image des contours en effectuant un seuillage de la norme du gradient ou des maximums locaux du gradient. On peut effectuer un seuillage standard en utilisant les fonctions de seuillage de l'activité précédente ou un seuillage par hystérésis dont l'algorithme est décrit diapositive 36.

Activité **6**

*Écrire le fichier `hysteresis.c` qui contient la fonction `hysteresis_thr` calculant le seuillage par hystérésis relativement à un seuil haut et un seuil bas (l'interface `hysteresis.h` est fournie). On pourra tester cette fonction avec le programme `test-6.c`. Des exemples de tests sont proposés à la fin de l'énoncé et les images correspondantes sont fournies.*

### Conseils et remarques

— attention, le fichier `test-6` utilise également les fonctions `threshold_std`, `threshold_mean` et `threshold_percent` de l'activité précédente.

# Tests

`test-1 :` convolution 2D, symmétrique antidiagonal, affichage image dérivative

- `test-1 -h -n mand.png mand-1-h-n.png`
- `test-1 -h -a mand.png mand-1-h-a.png`
- `test-1 -v -a -i mand.png mand-1-v-a_i.png`

`test-2 :` dérivées partielles avevc les filtres D2, D3, de Prewitt, de Sobel

- `test-2 -di -2 -a mand.png mand-2-di-2-a.png`
- `test-2 -di -3 -a mand.png mand-2-di-3-a.png`
- `test-2 -di -p -a mand.png mand-2-di-p-a.png`
- `test-2 -di -s -a mand.png mand-2-di-s-a.png`
- `test-2 -dj -s -a mand.png mand-2-dj-s-a.png`

`test-3 :` dérivées partielles avec le filtre de Deriche pour $\alpha = 0.7$ et $\alpha = 2$

- `test-3 -di -d 0.7 -a mand.png mand-3-di-d-07-a.png`
- `test-3 -dj -d 0.7 -a mand.png mand-3-dj-d-07-a.png`

- `test-3 -di -d 2 -a mand.png mand-3-di-d-2-a.png`
- `test-3 -dj -d 2 -a mand.png mand-3-dj-d-2-a.png`

`test-4 :` norme des gradients de Prewitt, Sobel et Deriche

- `test-4 -p mand.png mand-4-p.png`
- `test-4 -s mand.png mand-4-s.png`
- `test-4 -d 0.7 mand.png mand-4-d07.png`
- `test-4 -d 2 mand.png mand-4-d2.png`

`test-5 :` maximums locaux des gradients de Prewitt, Sobel et Deriche

- `test-5 -p mand.png mand-5-p.png`
- `test-5 -s mand.png mand-5-s.png`
- `test-5 -d 0.7 mand.png mand-5-d07.png`
- `test-5 -d 2 mand.png mand-5-d2.png`

`test-6 :` seuillages standards et par hystérésis de la norme et des maximums locaux des gradients de Sobel et Deriche

- `test-6 50 -i mand-5-s.png mand-6-s-th11.png`
- `test-6 140 -i mand-5-s.png mand-6-s-th20.png`
- `test-6 -h 50 140 -i mand-5-s.png mand-6-s-h11-20.png`

- `test-6 5 -i mand-5-d07.png mand-6-d07-th11.png`
- `test-6 10 -i mand-5-d07.png mand-6-d07-th20.png`
- `test-6 -h 5 10 -i mand-5-d07.png mand-6-d07-h11-20.png`

- `test-6 13 -i mand-5-d2.png mand-6-d2-th13.png`
- `test-6 30 -i mand-5-d2.png mand-6-d2-th30.png`
- `test-6 -h 13 30 -i mand-5-d2.png mand-6-d2-h13-30.png`