Exercices de Programmation & Algorithmique 1

Série 8 – Détection de contours
(16 novembre 2021)

Département d'Informatique – Faculté des Sciences – UMONS

Pré-requis : Avoir terminé les contrats précédents.

Objectifs : Résoudre un problème complet ; être capable de lire et comprendre un algorithme ; être capable de se renseigner sur un algorithme.

---

1 Contexte

La détection de contours est une technique visant à identifier sur une image les points qui correspondent à des changements importants de luminosité. Ces modifications de la luminosité sont généralement issus des changements de matériaux, d'orientations, de profondeur ou des éclairages de la scène ainsi représentée sur l'image.

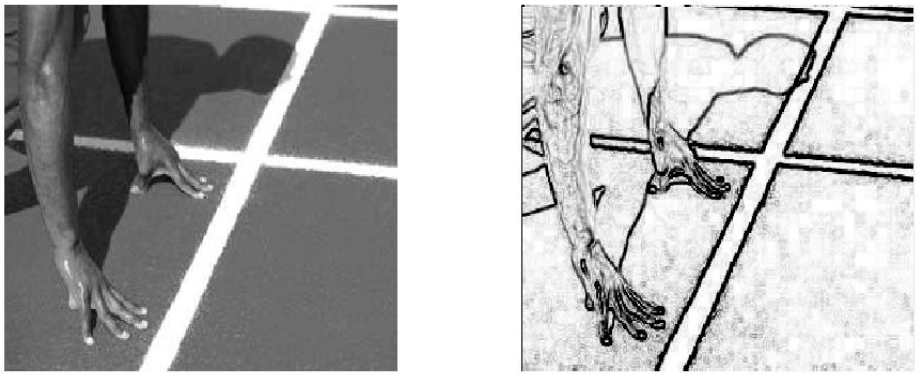

FIGURE 1 – Exemple d'une détection de contours.

Le principe de la détection de contours repose donc sur l'étude des dérivées de la fonction d'intensité dans l'image. L'intensité d'un pixel peut être la valeur de ce pixel une fois l'image transformée en niveau de gris. Il existe de nombreuses techniques pour détecter les contours dans une image. L'une d'elle, la plus simple, repose sur l'opérateur de Sobel utilisé comme fonction d'intensité.

L'opérateur de Sobel est une opération mathématique impliquant des matrices de convolution. Cet opérateur est utilisé sur chaque région de l'image afin de calculer le gradient (les "dérivées") d'intensité de chaque pixel de l'image. On obtient ainsi des informations sur les changements soudains de luminosité ainsi que de la direction de ces différents changements. Cette matrice, souvent de taille 3×3, subit une convolution avec l'image pour calculer des approximations des dérivées verticale et horizontale.

Une matrice de convolution est une matrice, généralement carrée et de petite taille, qui définit le traitement à appliquer point par point dans une autre matrice (par exemple, une image). La figure 2 donne un exemple d'une telle matrice 3×3 appliquée à *une* position d'une matrice d'entiers correspondant à une image donnée. Dans cet exemple, les éléments de la matrice de gauche correspondant aux éléments de la zone d'action de la matrice de convolution ont été lus de gauche à droite et de haut en bas. Chacun d'entre eux a été multiplié par la valeur correspondante et les différents

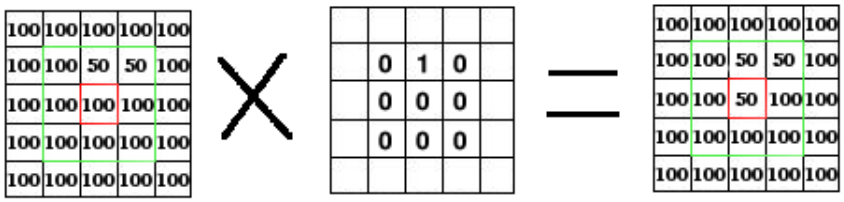

FIGURE 2 – Exemple de matrice de convolution 3×3.

FIGURE 3 – Exemple d'image de 4 pixels de large sur 6 pixels de haut.

résultats ont été additionnés. Ainsi, la valeur centrale de la matrice résultante est bien de 50 car $(100 \times 0) + (50 \times 1) + (50 \times 0) + (100 \times 0) + (100 \times 0) + (100 \times 0) + (100 \times 0) + (100 \times 0) = 50$. Une telle matrice de convolution correspond typiquement à une opération de décalage vers le bas.

2 Le contrat

Dans le cadre de ce contrat, un module `umage` est mis à votre disposition. Ce module, qui nécessite la libraire PIL de Python (Python Imaging Library), vous permet d'ouvrir un fichier image présent sur le disque et d'en obtenir une copie en mémoire sous la forme d'une liste (les lignes) de listes (les colonnes) de triplets RGB. Un triplet RGB est un triplet de la forme `(rouge, vert, bleu)` où chaque composante indique l'intensité lumineuse de la couleur correspondante. Cette intensité est un entier compris entre 0 (le moins lumineux) et 255 (le plus lumineux). Dans une image, accéder à l'élément en position (i, j) signifie accéder au j^{e} pixel horizontal de la i^{e} ligne.

La Figure 3 donne un exemple d'image de 4 pixels de large sur 6 pixels de haut. Cette image, une fois chargée par le module `umage`, correspond à la matrice suivante, encodée en Python :

```
[[(255, 255, 255), (30, 0, 255), (30, 0, 255), (0, 0, 0)],
 [(30, 0, 255), (0, 255, 0), (0, 255, 0), (255, 0, 0)],
 [(255, 0, 0), (30, 0, 255), (30, 0, 255), (255, 0, 0)],
 [(0, 255, 0), (30, 0, 255), (30, 0, 255), (255, 0, 0)],
 [(0, 0, 0), (0, 255, 0), (30, 0, 255), (255, 0, 0)],
 [(0, 0, 0), (0, 255, 0), (0, 255, 0), (0, 255, 0)]]
```

Notez bien qu'aucune des fonctions demandées dans ce contrat ne peut présenter d'effets de bord.

1 Écrivez la fonction `greyscale(mat_img)` qui retourne une image similaire à l'image `mat_img` en niveaux de gris. Un pixel en niveau de gris est un pixel pour lequel les trois composantes ont la même valeur. Cette valeur est obtenue grâce à la formule suivante : $V = 0.2125 \times R + 0.7154 \times G + 0.0721 \times B$. Un exemple d'entrée/sortie est illustré à la Figure 4.

2 Écrivez l'analyse et implémentez la fonction `convolution(mat_img, mat)` qui retourne une nouvelle image correspondant à l'application, sur chaque composante des pixels, de la matrice de convolution fournie par `mat` sur l'image `mat_img` donnée en paramètre. Vous pouvez considérer que la matrice de convolution est toujours carrée et de taille impaire. Vous pouvez considérer que les pixels en dehors de l'image ont pour valeur $(0, 0, 0)$. Notez que, pour vous aider :

— vous pouvez réaliser les exercices supplémentaires `pixel` et `appliquer_convolution` et,

— vous pouvez, **dans un premier temps**, considérer que la matrice de convolution est de taille 3×3.

De nombreux filtres basiques permettent d'obtenir de bons résultats juste en travaillant avec des matrices de convolution. À titre d'exemple, essayez de voir ce que font les matrices suivantes :

```
[[-1,-1,-1],[-1,8,-1],[-1,-1,-1]]
[[-1,-1,-1],[-1,9,-1],[-1,-1,-1]]
[[-2,0,0],[0,1,0],[0,0,2]]
```

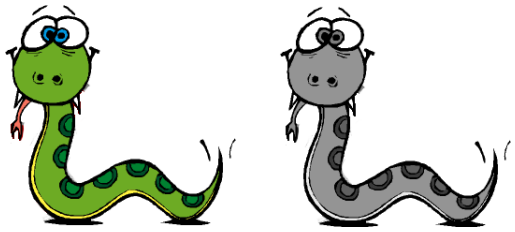

FIGURE 4 – Exemple d'entrée/sortie pour la fonction `greyscale`.

L'exercice suivant vous demande de vous renseigner et d'implémenter une détection de contours utilisant les idées de Sobel. Essayez d'identifier au mieux les différentes opérations nécessaires pour appliquer un tel filtre.

3 Renseignez-vous sur la terminologie et l'algorithme de Sobel [1].

4 Effectuez l'analyse Top-Down de l'algorithme de Sobel. Pensez à séparer les concepts distincts en différentes fonctions et à bien identifier, pour chacune d'elles, les entrées/sorties.

> **Important** : avant de passer à la suite du contrat (sur machine); vous devez faire valider vos solutions par un assistant ou élève-assistant.

5 Implémentez l'algorithme de Sobel sur machine. Un exemple de résultat à obtenir sur base de l'image test est repris dans la Figure 5.

FIGURE 5 – Exemple de résultat à obtenir sur l'image test.

3 Exercices complémentaires

★☆☆ 6 Écrivez une fonction booléenne `is_greyscale(img)` qui retourne la valeur booléenne `True` si et seulement si l'image passée en paramètre est une image en niveaux de gris.

1. Vous pouvez considérer `http://fr.wikipedia.org/wiki/Filtre_de_Sobel` comme point de départ.

★☆☆ 7 Écrivez une fonction `invert(img)` qui retourne le négatif de l'image passée en paramètre. Le négatif d'un pixel P est un pixel $P' = (255 - P_r, 255 - P_g, 255 - P_b)$.

★☆☆ 8 Écrivez une fonction `pixel(img, i, j, default)` qui retourne le pixel en position (i, j) de l'image donnée en paramètre si ce pixel fait partie de l'image, la valeur de `default` dans le cas contraire.

★★☆ 9 Écrivez une fonction `appliquer_convolution(img, mat, i, j)` qui calcule la valeur retournée par l'application de la matrice de convolution `mat` appliquée sur `img` à la position (i, j) (précondition).

★★☆ 10 Écrivez une fonction booléenne `is_symmetrical(img)` qui retourne la valeur booléenne `True` si et seulement si l'image passée en paramètre possède un axe de symétrie orthogonal horizontal ou vertical.

★★☆ 11 Écrivez une fonction `circle(img, (cx, cy), radius, color)` qui dessine, dans l'image donnée en paramètre, un cercle de centre (c_x, c_y), de rayon `radius` (exprimé en pixels) et de couleur `color` (un triplet RGB).

★★☆ 12 Écrivez une fonction `flou(img, r)` qui retourne une nouvelle image correspondant à l'application d'un flou de rayon `r` sur l'image `img` (ce flou calcule pour chaque pixel de l'image de départ la moyenne des valeurs des pixels présents dans un cercle de rayon `r`). Un exemple d'entrée/sortie est illustré à la Figure 6. Assurez-vous que le flou produise un résultat lisse, et non trouble.

FIGURE 6 – Exemple d'entrée/sortie pour la fonction `flou` avec un rayon de 5.

★★★ 13 La détection de contour par filtre de Canny :

Une variante de détection de contours réputée pour sa vitesse d'exécution est l'utilisation d'un filtre de Canny. Cette variante fait intervenir d'autres filtres (par exemple, un flou gaussien) afin d'améliorer nettement la détection des contours et d'éviter au maximum les faux positifs. Ainsi, les variations d'intensité sont non seulement prises en compte mais également la (ou les) direction(s) de ces variations permettant d'éliminer de faux contours qui ne correspondent pas à des véritables changements d'intensité lumineuse.

— Renseignez-vous sur la terminologie et l'algorithme de Canny.
— Effectuez l'analyse Top-Down de l'algorithme de Canny.
— Implémentez l'algorithme de Canny sur machine.