Algorithmique des images TD n° 2

Détection de contours

Le but de ce TP est d'implanter les différents détecteurs de contours vus en cours (naïf, Sobel, Canny).

Dans ce TP on suppose que l'on utilisera les structures `pgm_t` `ppm_t` et définie dans les TP précédents. On se restreindra dans un premier temps à des images en nuances de gris (`.pgm`).

Exercice 1. Détecteurs naïf et de Sobel

Q- 1.1 Item 1

Créer une fonction `unsigned char max_pgm(pgm_t *image)` qui renvoie la valeur maximale des pixels de l'image `image` passée en paramètre.

Q- 1.2 Item 2

Créer des fonctions `pgm_t *naive_x(pgm_t *image)` et `pgm_t *naive_y(pgm_t *image)` qui renvoient l'image correspondant respectivement au détecteur de contours horizontaux et verticaux.

`Remarque` : ces détecteurs peuvent être vus comme la convolution de l'image avec les noyaux 3×3 suivants :

$$\mathrm{K_h} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad K_v = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Q- 1.3 Item 3

Créer une fonction `pgm_t *naive_edge_detector(pgm_t *image)` permettant d'implanter le détecteur de contours naïf.

`Remarque` : on fera attention à normaliser la valeur des pixels dans $[0, 255]$ en calculant :

$$P(i,j) = \left\lfloor P(i,j) \cdot \frac{255}{M} \right\rceil$$

avec M la valeur maximale des pixels $(P(i,j))_{i,j}$ de la norme du vecteur $[naive\_x, naive\_y]$.

Q- 1.4 Sobel

En suivant le même raisonnement créer des fonctions `pgm_t *sobel_edge_detector(pgm_t *image)` implantant le filtre de Sobel.

Exercice 2. Détecteur de Canny

Q- 2.1 Gaussian

Créer une fonction `void gaussian_blur(pgm_t *image, double sigma, int n)` qui applique un filtre gaussien de taille $n = 2p + 1$ et d'écart type σ à l'image `image` passée en paramètre.

`Rappel` : pour appliquer un filtre gaussien à une image on calcule sa convolée avec le noyau $(2p+1)\times(2p+1)$ $K_G = (G_\sigma(i-p)\cdot G_\sigma(j-p))_{0\leq i,j\leq 2p}$ avec :

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

Q- 2.2 Gradient

Créer une fonction `double **gradiant_angle(pgm_t *grad_x, pgm_t *grad_y)` qui prend en paramètre les deux composantes du gradient et renvoie l'angle de celui ci avec la tangeante horizontale. On utilisera la fonction `atan2` de la bibliothèque `math.h`

Q- 2.3 Maxima

Créer une fonction `void non_maxima_suppression(pgm_t *norm, double **angle)`, la fonction prendra en paramètre une image correspondant à la norme du gradient, et un tableau 2D de `double` contenant l'angle du gradient avec la droite horizontale pour chaque pixel de l'image (sauf les bords).

`Remarque` : on fera attention à copier les valeurs de l'image `norm` passée en paramètre avant de supprimer les non-maximaux.

Afin de pouvoir implémenter le seuillage par hystéresis, nous devons lister les pixels étant des contours certains. Étant donné que l'on ne peut pas connaitre leur nombre à l'avance, nous allons utiliser une structure de liste chaînée.

Q- 2.4 Pile

Créer une structure de pile en liste chaînée avec des fonctions pour ajouter et supprimer des éléments.

Q- 2.5 Hysteresis

Créer une fonction `void hysteresis_thresholding(pgm_t *image, float seuil_haut, float seuil_bas)` implantant le double seuillage avec traquage par hystéresis. Les paramètres `seuil_haut` et `seuil_bas` seront des valeurs entre 0 et 1 représentant le pourcentage de la valeur maximale considéré.

Q- 2.6 Canny

Créer `pgm_t *canny_edge_detector(pgm_t *image, double sigma, int n)` implantant le filtre de Canny avec un lissage gaussien de taille `n` et d'écart type `sigma`.

Q- 2.7 Format RGB

Modifier vos fonctions précédentes de sorte à prendre également en charge des images avec des pixels en `RGB` – i.e. au format `ppm`.

Q- 2.8 Main

Écrire un programme principal qui prendra en paramettre une image `pgm` ou `ppm`, le type de filtre à appliquer (naïf, Sobel, Canny) et pour le cas du filtre Canny l'utilisateur passera en plus les paramètres `sigma`, la taille `n` du filtre est les valeurs des seuils haut et bas. Le programme stockera le résultat de la détection de contours dans un nouveau fichier.