PROGRAMMATION C
TP 9 - TRAITEMENT D'IMAGE (3)

LICENCE MATHS-INFO
17 AVRIL 2012

Dans ce dernier TP, vous reprendrez votre code du TP 8 et le complèterez en y ajoutant de nouveaux effets. Vous trouverez sur le site du module de nouvelles images.

Égalisation d'histogramme

L'égalisation d'histogramme permet de répartir l'intensité des niveaux de gris de façon équilibrée. Cela permet d'ajuster le contraste pour par exemple traiter des images sous- ou sur-exposées.

On définit l'histogramme sur L niveaux d'une image comme un tableau h de L entiers ; pour $0 \leq k < L$, la k-ième case est le nombre de pixels de l'image qui ont une valeur x est telle que $k \leq x \times (L-1) < k+1$. Autrement dit, k est la partie entière de $x \times (L-1)$.

L'histogramme de l'image `siberie.pgm` est représenté sur la partie gauche de la figure 1, avec $L = 256$. On voit que les niveaux de gris sont concentrés sur une zone restreinte, ce qui explique le manque de contraste de l'image. L'égalisation consiste à transformer l'image pour utiliser toute l'étendue des niveaux de gris, comme illustré sur la partie droite de la figure 1.

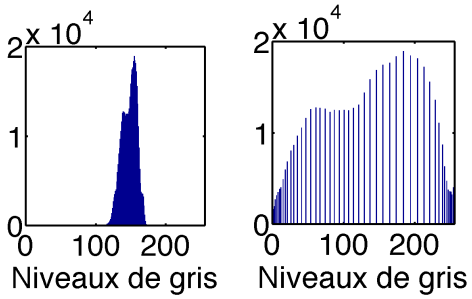

Image originale Après égalisation

FIGURE 1. Histogrammes avant et après égalisation.

On construit un second tableau de taille L, l'histogramme cumulé H, défini par

$$H[k] = \sum_{j=0}^{k} h[j].$$

Ensuite, on construit la nouvelle image pixel par pixel. Pour un pixel de valeur x dans l'image originale, le pixel correspondant dans la nouvelle image a pour valeur $\frac{H[k]}{n}$, où n est le nombre de pixels dans l'image et k est tel que $k \leq x \times (L-1) < k+1$.

Écrivez une fonction

`Image *egalisation(Image *p_image, int L)`

réalisant l'égalisation d'histogramme sur `L` niveaux de l'image pointée par `p_image` et renvoie un pointeur sur la nouvelle image. Cette fonction réalisera successivement la construction de l'histogramme, de l'histogramme cumulé et de la nouvelle image. Vous testerez cet effet sur `siberie.pgm` et `temple.pgm` avec `L=256`.

Morphologie mathématique

La morphologie mathématique est un cadre théorique et applicatif qui permet de nombreux effets en traitement d'images. En voici quelques-uns obtenus à partir de deux fonctions de base, l'érosion et la dilatation.

Notion d'élément structurant. Un élément structurant est défini comme une matrice de même dimension `PATCH_DIM`×`PATCH_DIM` qu'un patch et dont les éléments sont soit 0, soit 1. Vous testerez les éléments structurants suivants :

- le carré : tous les coefficients à 1.
- le croix : tous les coefficients à 1 sur la colonne centrale et la ligne centrale.
- le triangle : tous les coefficients à 1 sur une diagonale et au-dessus.

Définissez ces éléments comme des variables globales `ES_carre`, `ES_croix` et `ES_triangle`. Vous pouvez écrire une fonction `void init_ES()` qui initialise ces matrices et est appelée une fois au début du programme.

Érosion et dilatation. L'érosion et la dilatation sont des traitements par patch. Pour un patch donné, on sélectionne les éléments du patch pour lesquels l'élément structurant vaut 1 à la position correspondante et on renvoie le minimum (resp. le maximum) de ces éléments pour obtenir une érosion (resp. une dilatation).

Définissez les fonctions suivantes :

- `double erosion_pix(double **patch, int es[PATCH_DIM][PATCH_DIM])` qui calcule l'érosion pour un patch et un élément structurant donnés.
- `double erosion_carre_pix(double **patch)` qui calcule, en une ligne, l'érosion pour un patch avec l'élément structurant « carré ». Idem pour `erosion_croix_pix` et `erosion_triangle_pix`.
- `Image *erosion_carre(Image *p_image)` qui calcule, en une ligne, l'érosion d'une image avec l'élément structurant « carré ». Idem pour `erosion_croix` et `erosion_triangle`.

Faire la même chose pour la dilatation. Testez ces fonctions sur `number72.pgm`

Effets à réaliser. Vous pouvez maintenant facilement faire les effets suivants :

- Ouverture : une érosion suivie d'une dilatation.
- Fermeture : une dilatation suivie d'une érosion.
- Détection de contour 1 : calcul d'une image dilatée D, d'une image érodée E, puis du résultat R par $\forall i, j, R_{ij} = D_{ij} - E_{ij}$.
- Dét. de contour 2 : idem avec $\forall i, j, R_{ij} = D_{ij} + E_{ij} - 2I_{ij}$ où I est l'image initiale.

Pour la détection de contour, vous forcerez à 0 (resp. 1) les pixels qui seraient de valeurs négatives (resp. supérieures à 1). Vous pouvez tester ces opérations avec les différents éléments structurants sur les images `number72.pgm` et `lacornouaille.pgm`.