Université Louis Pasteur
UFR de Mathématique et Informatique

Session de Juin 2002
Maîtrise d'informatique

Traitement d'Images

Corrigé
par : Prof. Christian RONSE

(1) Seuillage percentile *(4 points)*

On suppose que l'intevalle de niveaux de gris va de 0 à M. Soit n la taille (le nombre de pixels) de l'image. Nous écrivons C_I pour l'histogramme cumulatif de l'image I, c.à.d. pour tout niveaux de gris g, $C_i(g)$ est le nombre de pixels dont le niveau de gris dans I est inférieur ou égal à g (i.e., le nombre de pixels p tels que $I(p) \leq g$). Donc $C_I(g)/n$ donne la *proportion* de pixels de l'image I dont le niveau de gris est inférieur ou égal à g.

Nous appliquons à l'image I l'égalisation d'histogramme ; celle-ci réalise la transformation de niveaux de gris $g \mapsto M \cdot C_I(g)/n$ (arrondi à un entier). Soit I' l'image égalisée. Nous appliquons ensuite à I' un seuillage où les pixels avec $I'(p) > 0,7 \cdot M$ deviennent blancs, et ceux avec $I'(p) \leq 0,7 \cdot M$ deviennent noirs.

On a ainsi : p devient blanc ssi $I'(p) > 0,7 \cdot M$, ssi $M \cdot C_I\big(I(p)\big)/n > 0,7 \cdot M$ (ceci est approximativement vrai, à cause du problème d'arrondi de $I'(p)$), ssi $C_I\big(I(p)\big)/n > 0,7$; réciproquement, p devient noir ssi $C_I\big(I(p)\big)/n \leq 0,7$ (tout aussi approximativement). Cela donne une bonne réponse au problème. En effet, comme l'image est discrète, il n'est généralement pas possible de trouver un seuil de niveau de gris en dessous duquel on a *exactement* 70% des pixels ; on aura plutôt un seuil faisant passer $C_I(g)$ de $< 70\%$ à $> 70\%$. Ici la proportion de pixels devenant blancs sera la plus petite possible qui soit $\geq 30\%$, et celle de pixels devenant noirs la plus grande possible qui soit $\leq 70\%$.

Notez que si on veut un résultat plus précis, non tributaire des problèmes d'arrondi, au lieu d'utiliser un tel logiciel, on fera plutôt un programme calculant l'histogramme cumulatif et trouvant le seuil par comparaison des $C_I(g)$ avec $0,7 \cdot n$.

(2) Points simples *(3 points)*

Soit p le pixel central de la configuration.

(a) En (a) on a

- (4) $Y_4(p) = 1$ (p est 4-adjacent à 1 composante 4-connexe de pixels de la figure dans son 8-voisinage, et 8-adjacent à 1 composante 8-connexe de pixels du fond dans son 8-voisinage).
- (8) $Y_8(p) = 2$ (p est 8-adjacent à 2 composantes 8-connexes de pixels de la figure dans son 8-voisinage, et 4-adjacent à 2 composantes 4-connexes de pixels du fond dans son 8-voisinage).

(b) En (b) on a

- (4) $Y_4(p) = 2$ (p est 4-adjacent à 2 composantes 4-connexes de pixels de la figure dans son 8-voisinage, et 8-adjacent à 2 composantes 8-connexes de pixels du fond dans son

8-voisinage).

(8) $Y_8(p) = 1$ (p est 8-adjacent à 1 composante 8-connexe de pixels de la figure dans son 8-voisinage, et 4-adjacent à 1 composante 4-connexe de pixels du fond dans son 8-voisinage).

Donc p est simple en $(a)(4)$ et en $(b)(8)$, mais non simple en $(a)(8)$ et en $(b)(4)$.

NB. On a la symétrie par interversion de (a) et (b) et de (4) et (8). En effet, si on intervertit 1 et 0 (pixels de la figure et du fond) dans le 8-voisinage de p, la configuration (a) donne (b) à une rotation d'un quart de tour près.

(3) Filtre linéaire, médian, ou de Kramer-Bruckner *(5 points)*

L'image en demi-ton I_d comporte des pixels noirs et blancs, et dans une petite zone la proportion entre les nombres de pixels blancs et de pixels noirs dans I_d correspond au niveau de gris des pixels de cette zone dans l'image originelle I (en supposant que ce niveau de gris varie peu dans cette zone). Cela est relativement théorique, les algorithmes de demi-ton ne sont pas parfaits.

Lissage linéaire: Sur une zone à niveau de gris constant, dans toute fenêtre 3×3 la proportion entre les nombres de pixels blancs et de pixels noirs dans I_d correspond au niveau de gris des pixels de cette zone dans I, et ainsi la moyenne dans la fenêtre (que réalise le lissage linéaire) devrait (en théorie) donner le niveau de gris dans I. Dans une zone à variations de niveaux de gris graduelles (rampe), cela restera plus ou moins vrai. Dans le cas de variation brusque, par exemple une arête, pour un pixel voisin de cette variation, la moyenne des niveaux de gris dans la fenêtre mélangera les proportions de pixels blancs et noirs correspondant à deux plages d'intensités différentes dans l'image originelle I, donc il y aura un peu de flou par rapport à I. En conclusion, le lissage linéaire résout relativement bien le problème, mais introduit du flou (comme toujours).

NB. Si l'algorithme de dithering utilise des voisinages plus grands que 3×3, il faudra aussi utiliser un masque plus grand pour le lissage linéaire. Bien sûr le flou augmente avec la taille du masque.

Filtre médian: Comme l'image I_d est binaire (les pixels y sont soit tout noirs soit tous blancs), le filtre médian effectue un “vote majoritaire” dans la fenêtre, de la valeur noire ou blanche à affecter au pixel de référence. Donc l'image restera binaire. Dans une zone dont le niveau de gris dans l'image originelle I était sombre (c.à.d. inférieur à la moyenne entre blanc et noir), la fenêtre 3×3 autour d'un pixel contiendra une majorité de pixels noirs dans I_d, et le pixel deviendra noir ; donc une telle zone sombre deviendra noire. Semblablement, une zone dont le niveau de gris dans l'image originelle I était clair (c.à.d. supérieur à la moyenne entre blanc et noir), deviendra blanche. Le long d'une arête, l'affectation blanc ou noir peut être déviée.

On peut aussi remarquer que pour des valeurs binaires blanc et noir, la médiane (“vote majoritaire”) vaudra blanc ssi il y a une majorité de blanc, c.à.d. ssi la moyenne des valeurs est supérieure à la moyenne entre blanc et noir (gris moyen). Donc sur une image binaire, le filtre médian correspond à un seuillage du lissage linéaire, de seuil égal à la moyenne entre les deux valeurs binaires. Donc ici on obtient à peu près un seuillage d'un flou de l'image originelle.

Kramer-Bruckner: Comme l'image I_d est binaire, l'opérateur de Kramer et Bruckner ne modifie pas celle-ci. En effet, un pixel blanc (resp. noir) aura son niveau de gris égal au maximum (resp. minimum) des niveaux de gris de sa fenêtre, qui est donc l'extrémum le plus proche qui sera donné comme niveau de gris au pixel.

Conclusion : Il faut préférer le lissage linéaire, malgré le léger flou. On peut facilement faire l'expérience sous XV : d'abord on sauvegarde une image en mettant pour la couleur "B/W Dithered", et sur cette image en demi-ton on peut essayer les algorithmes "Blur" (lissage linéaire) et "Despeckle" (filtre médian).

(4) Composantes connexes *(3 points)*
Le nombre d'Euler en k-connexité pour la figure et k'-connexité pour le fond est $E_k = C_k(F) - C_{k'}(B) + 1$. On a donc

$$E_4 - E_8 = \big[C_4(F) - C_8(B) + 1\big] - \big[C_8(F) - C_4(B) + 1\big] = C_4(F) - C_8(F) + C_4(B) - C_8(B) \ .$$

On peut alors utiliser les formules

$$E_4 = \frac{Q_1 - Q_3 + 2X}{4} \qquad \text{et} \qquad E_8 = \frac{Q_1 - Q_3 - 2X}{4} \ ,$$

où Q_1 est le nombre de configurations 2×2 avec 1 pixel de la figure et 3 du fond, Q_3 est le nombre de configurations 2×2 avec 3 pixels de la figure et 1 du fond, et X est le nombre de configurations 2×2 avec 2 pixels de la figure sur une diagonale et 2 du fond sur l'autre. Cela donne

$$E_4 - E_8 = \frac{Q_1 - Q_3 + 2X}{4} - \frac{Q_1 - Q_3 - 2X}{4} = X \ ,$$

d'où finalement

$$C_4(F) - C_8(F) + C_4(B) - C_8(B) = E_4 - E_8 = X \ .$$

(5) Algorithme séquentiel *(7 points)*
(i) L'initialisation donne :

$$\begin{matrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & 0 & \infty \\ \infty & 0 & \infty & \infty & \infty & 0 & \infty \\ \infty & 0 & 0 & 0 & 0 & 0 & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{matrix}$$

Ensuite, les deux balayages horizontaux de gauche à droite et de droite à gauche donnent :

$$\begin{matrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & 2 & 3 & 0 & 1 \\ \infty & 0 & 1 & 2 & 3 & 0 & 1 \\ \infty & 0 & 0 & 0 & 0 & 0 & 1 \\ \infty & \infty & \infty & 0 & 1 & 2 & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{matrix} \qquad \begin{matrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{matrix}$$

Enfin, les deux balayages verticaux de haut en bas et de bas en haut donnent :

$$\begin{matrix} \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 1 & 1 & 2 \\ 3 & 2 & 2 & 1 & 2 & 2 & 3 \end{matrix} \qquad \begin{matrix} 2 & 1 & 2 & 3 & 2 & 1 & 2 \\ 1 & 0 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 1 & 1 & 2 \\ 3 & 2 & 2 & 1 & 2 & 2 & 3 \end{matrix}$$

(ii) L'exemple ci-dessus montre que cet algorithme réalise la transformée de distances pour la 4-distance d_4. Expliquons-le d'une manière générale.

Lors du premier balayage de gauche à droite, chaque fois qu'on sort de X, les pixels à droite de X recevront successivement les valeurs $1, 2, 3, \ldots$, jusqu'à ce qu'on se retrouve à nouveau dans X. Donc la valeur en un pixel sera sa distance horizontale au plus proche pixel de X situé à gauche de celui-ci.

Durant le second balayage de droite à gauche, chaque fois qu'on sort de X, les pixels à gauche de X recevront successivement les valeurs $1, 2, 3, \ldots$, sauf s'ils ont une valeur inférieure (provenant du premier balayage). Donc la valeur en un pixel sera sa distance horizontale au plus proche pixel de X, lequel est situé à gauche de celui-ci si la valeur a été obtenue au premier balayage, et à droite si elle a été obtenue au second balayage.

Au troisième balayage, après passage sur un pixel avec la valeur n, les pixels suivants (de haut en bas) recevront successivement les valeurs $n + 1, n + 2, n + 3, \ldots$, sauf s'ils ont une valeur inférieure (provenant d'un balayage antérieur) ; dans ce cas le pixel garde sa valeur m, et on reprend l'affectation itérative de valeurs $m + 1, m + 2, m + 3, \ldots$ sous condition de ne pas avoir une valeur inférieure (obtenue auparavant). Donc la valeur en un pixel p sera le plus petit nombre $h + v$, tel qu'il y a un pixel q au-dessus de p, à distance v de p, et tel que la distance horizontale de q à X (sa valeur obtenue à l'issue du deuxième balayage) soit h.

Finalement au quatrième balayage, après passage sur un pixel avec la valeur n, les pixels suivants (de bas en haut) recevront successivement les valeurs $n + 1, n + 2, n + 3, \ldots$, sauf s'ils ont une valeur inférieure, et dans ce cas le pixel garde sa valeur et on recommence l'itération. Donc la valeur en un pixel p sera le plus petit nombre $h + v$, tel qu'il y a un pixel q sur la même colonne que p, à distance v de p, et tel que la distance horizontale de q à X soit h ; q sera au-dessus de p ou en dessous, suivant que la valeur en p aura été obtenue au troisième ou au quatrième balayage. Ce nombre $h + v$ est la longueur du plus court chemin de X à p formé d'un segment horizontal et d'un segment vertical, donc c'est la 4-distance de p au plus proche pixel de X, c.à.d. $d_4(p, X)$.

Remarque: On pourrait donner une preuve beaucoup plus formelle (utilisant une induction pour chacun des 4 balayages, donc très longue), mais ce n'était pas l'objet de l'exercice d'aller aussi loin.

NB. Cet algorithme a été proposé par Fernand Meyer en 1987, dans un communication intitulée “Algorithmes séquentiels”.