Informatique 1ere année, CPBX, TD4

Carole Blanc, Paul Dorbec

1 Ouvrons le placard

Pour cet exercice, vous utiliserez l'image `placard.png`.

Question 1.1 *On souhaite ouvrir le placard. Pour cela, vous prendrez la porte de gauche (des pixels* $(71, 4)$ *à* $(131, 186)$*) en la décalant de 64 pixels vers la gauche. Vous pourrez remplir l'espace libéré avec du noir, ou avec l'image* `fantome.png`*.*

Question 1.2 *Même question en entrouvrant la porte de droite (des pixels* $(135, 4)$ *à* $(196, 186)$*) en la décalant de seulement 32 pixels vers la droite (attention il peut y avoir un piège !).*

2 Niveaux de couleur

Pour refaire les niveaux d'une image, on peut utiliser la méthode suivante :

— pour chaque couleur, on cherche la valeur minimum v_m et la valeur maximum v_M apparaissant parmi les pixels de l'image, on obtient la gamme de cette couleur : l'intervalle $[v_m, v_M]$

— puis, on réaffecte à chaque pixel une nouvelle valeur v_N sur cette couleur, calculée en projetant l'intervalle de départ sur l'ensemble des valeurs disponibles $[0, 255]$. On obtient cette valeur à partir de la valeur initiale v_I en calculant

$$v_N = \frac{255(v_I - v_m)}{v_M - v_m}$$

Question 2.1 *Écrivez une fonction* `minimum_r(img)` *qui calcule la valeur minimale des coefficients de rouge sur toute l'image. Écrivez de même une fonction* `maximum_r(img)`*. Modifiez les fonctions pour qu'elles renvoient les minimum et maximum de chaque couleur.*

Question 2.2 *Écrivez le programme qui refait les niveaux de couleur de l'image avec la méthode précédente. Vous pourrez tester votre programme en déchiffrant l'image* `hell.png`*.*

3 Détection de contours

Pour détecter un contour, on applique à chaque pixel de l'image une transformation qui dépend des pixels voisins.

Question 3.1 *Écrivez une fonction* `contour_V(img)` *qui crée une nouvelle image dont la valeur de chaque pixel* (x, y) *est égale pour chaque couleur à la valeur* $2v(x, y) - v(x - 1, y) - v(x + 1, y)$ *(où* $(v(x, y)$ *indique la valeur du pixel* x, y *sur l'image initiale).*

Question 3.2 *Écrivez de même la fonction* `contour_H(img)` *qui crée une nouvelle image dont la valeur de chaque pixel* (x, y) *est égale pour chaque couleur à la valeur* $2v(x, y) - v(x, y - 1) - v(x, y + 1)$.

Question 3.3 *Calculez l'image dont la valeur de chaque pixel est la norme 2 des deux valeurs précédentes* ($\sqrt{a^2 + b^2}$)*. Testez sur une image de votre choix.*

4 Symétries

Question 4.1 *Écrivez une fonction* `miroir(img)` *qui prend une image et en échange la gauche et la droite. Testez.*

Question 4.2 *Écrivez une fonction* `renverser(img)` *qui fait effectuer à l'image une rotation de* 180°*. Testez.*

5 Crypter une image

Question 5.1 *Écrire une fonction* `bloc(img,x1,y1,x2,y2)` *qui prend en entrée une image* `img` *et les coordonnées d'un bloc rectangulaire de pixels dans cette image.* `(x1,y1)` *sont les coordonnées du pixel en haut à gauche du bloc et* `(x2-1,y2-1)` *celles du pixel en bas à droite du bloc. Cette fonction calcule la moyenne des couleurs des pixels du bloc entre* `(x1,y1)` *et* `(x2-1,y2-1)` *(inclus) et donne cette couleur à tous les pixels du bloc. On s'autorisera à modifier l'image en argument.*

Question 5.2 *Écrire une fonction* `crypte(img,k)` *qui traite l'image par bloc de taille* $k \times k$ *et remplace les pixels de chaque bloc par la moyenne de la couleur des pixels du bloc. On s'autorisera à modifier l'image en argument, mais on prendra bien soin de ne pas manipuler des pixels en dehors de l'image.*

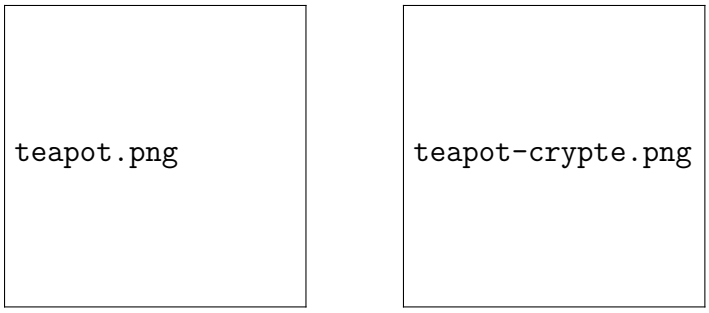

FIGURE 1 – `teapot` et `crypte(teapot,10)`