Licence de Sciences et Technologies — Université de Perpignan Via Domitia
S4 – mention Informatique — V. Zucca

TP4 Algorithmique des images, textes et données

Transformations géométriques d'une image.

Le but de ce TP est d'implanter des fonctions permettant d'appliquer les transformations géométriques de base d'une image.

Dans ce TP on suppose que l'on utilisera la structure `picture` définie dans le TP précédent. L'ordre d'interpolation sera défini comme une macro dans notre code :

```
#define N 1 // Interpolation d'ordre 1
```

On rappelle ci dessous les expressions des fonctions de poids nécessaire à l'interpolation :

$$\text{ordre } 0 \quad \omega_0(x) = \begin{cases} 0 & \texttt{si } |x| > 0.5 \\ 1 & \texttt{si } |x| < 0.5 \\ 0.5 & \texttt{si } |x| = 0.5 \end{cases}$$

$$\text{ordre } 1 \quad \omega_1(x) = \begin{cases} 0 & \texttt{si } |x| > 1 \\ x+1 & \texttt{si } -1 \leq x \leq 0 \\ 1-x & \texttt{si } 0 \leq x \leq 1 \end{cases}$$

$$\text{ordre } 2 \quad \omega_2(x) = \begin{cases} 0 & \texttt{si } |x| > 1.5 \\ 0.5(x+1.4)^2 & \texttt{si } -1.5 \leq x \leq -0.5 \\ 0.75 - x^2 & \texttt{si } -0.5 \leq x \leq 0.5 \\ 0.5(x-1.5)^2 & \texttt{si } 0.5 \leq x \leq 1.5 \end{cases}$$

$$\text{ordre } 3 \quad \omega_3(x) = \begin{cases} 0 & \texttt{si } |x| > 2 \\ 1/2|x|^3 - x^2 + 2/3 & \texttt{si } 0 \leq |x| \leq 1 \\ 1/6(2-|x|)^3 & \texttt{si } 1 \leq |x| \leq 2 \end{cases}$$

1. Créer des fonctions `double W0(double x)`, `double W1(double x)`, `double W2(double x)` et `double W3(double x)` qui renvoie les valeurs respectives de $\omega_0(x)$, $\omega_1(x)$, $\omega_2(x)$ et $\omega_3(x)$.

On définit en variable globale un tableau de pointeurs sur fonctions `W` de sorte à pouvoir appeler la bonne fonction ω selon la valeur de N.

```
double (*W[4])(double) = {W0, W1, W2, W3};
```

Ainsi en appelant `W[N]` on appellera la fonction de poids pour l'interpolation à l'ordre `N`.

2. Créer une fonction `unsigned char interpolation_pgm(picture *image, double x, double y)` qui renvoie la valeur interpolée du pixel en nuance de gris de coordonées `(x,y)` dans l'image `image`.
3. Créer une fonction `rgb interpolation_ppm(picture *image, double x, double y)` qui renvoie la valeur interpolée du pixel en rgb de coordonées `(x,y)` dans l'image `image`.
4. Créer des fonctions `picture *rotation_pgm(picture *image, double theta, int x0, int y0)` et `picture *rotation_ppm(picture *image, double theta, int x0, int y0)` qui calculent les rotations d'angle `theta` et de centre le point de coordonnées `(x0,y0)` des images au format `pgm` et `ppm` passées en paramètres.
5. Créer une fonction `picture *rotation(picture *image, double theta, int x0, int y0)` qui calcule la rotations d'angle `theta` et de centre le point de coordonnées `(x0,y0)` de l'image passée en paramètre.

En suivant le modèle précédent, créer les fonctions :

— `picture *zoom(picture *image, double lambda, int x0, int y0, int Dx, int Dy)` qui crée une image de taille `Dx x Dy` correspondant au zoom de facteur `lambda` autour du point de coordonnées `(x0,y0)` de l'image passée en paramètre.

— `picture *shear(picture *image, double lambda, int cx, int cy, int Dx, int Dy)` qui crée une image de taille `Dx x Dy` correspondant au cisaillement de facteur `cx`, `cy` de l'image passée en paramètre.