TP2-2013. Cisaillement et rotation d'images sous MATLAB

Cette seconde feuille de TP du cours 2013 illustre les cours dédiés au transformaions géométriques simples (cisaillements, rotations) ainsi qu'à une première sensibilisation au filtrage sous `MATLAB`.

Vous pourrez utiliser comme image-test par exemple l'image `bordeaux.jpg` ou bien l'image `cameraman.jpg` qui est celle utilisée pour illustrer le cours.

Exercice 1 (Cisaillements d'images). On considère dans cet exercice une image discrète noir et blanc (échantillonnée sur N niveaux de gris), codée au format flottant ramené à $[0,1]$, la grille de pixels étant $\{0,...,n-1\} \times \{0,...,m-1\}$ (j désigne l'indice de ligne et k l'indice de colonne, comme dans le cours). Le pas h est donc ici normalisé à 1 et l'image traitée est donc de taille $n \times m$ (m lignes, n colonnes).

(1) Soit $a \in \mathbb{R}$. Quelle est l'image par la transformation linéaire

$$\begin{bmatrix} x \\ y \end{bmatrix} \longmapsto \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

du cadre $[0, n-1] \times [0, m-1]$ de l'image initiale ? Même question avec la transformation linéaire

$$\begin{bmatrix} x \\ y \end{bmatrix} \longmapsto \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Quelle est dans ces deux cas le plus petit rectangle $[A_a, B_a] \times [0, m-1]$ (premier cas) ou $[0, n-1] \times [A_a, B_a]$ (deuxième cas) contenant l'image par la transformation linéaire du cadre $[0, n-1] \times [0, m-1]$?

(2) Rédigez des routines `cisaillhor1` et `cisaillver1`

```
>> II = cisaillhor1(I,a);
>> II = cisaillver1(I,a);
```

permettant le cisaillement d'une image horizontalement ou verticalement suivant le paramètre $a \in \mathbb{R}$. Distinguez pour cela les cas $a > 0$, $a = 0$ et $a < 0$. Vous serez dans tous les cas amenés à remplir une image dont la taille a été définie a priori par les dimensions des rectangles introduits à la question 1. On utilisera dans cette question l'approximation au point le plus proche (`round`) de la grille des pixels.

(3) Modifiez les deux routines que vous venez de réaliser en des routines `cisaillhor2` et `cisaillver2`

```
>> II = cisaillhor2(I,a);
>> II = cisaillver2(I,a);
```

réalisant les mêmes opérations que précédemment, mais exploitant cette fois le principe de l'approximation bilinéaire dans la grille des pixels.

EXERCICE 2 (rotation d'images).

(1) Vérifiez la formule établie dans le cours :

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix} = \begin{bmatrix}1 & 0\\ -\tan(\theta/2) & 1\end{bmatrix}\begin{bmatrix}1 & \sin\theta\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ -\tan(\theta/2) & 1\end{bmatrix} \qquad \forall\theta\in]-\pi,\pi[.$$

(2) Déduisez la construction de routines

```
>> J=rotation1(I,theta);
>> J=rotation2(I,theta);
```

opérant la rotation de l'image I de θ dans le sens trigonométrique.

(3) Quelle taille d'image faut-il prendre pour que toutes les rotations d'une image donnée de taille (m,n) s'effectuent dans le cadre imparti à l'image ? En utilisant le `zeropadding`, modifiez les codes établis à la question **2** (en deux routines `rotation1crop` et `rotation2crop`) de manière à ce que la taille de l'image de sortie soit celle de l'image d'entrée (comme dans l'option `'crop'` de la routine `imrotate`) et qu'aucune information concernant l'image ne se trouve « rognée » lors du processus de rotation.

EXERCICE 3 (rotation d'une image *via* la transformation bilinéaire et sans recours au cisaillement). On considère une image `I` de taille (m,n), vivant dans le cadre $[0,n-1]\times[0,m-1]$ et représentée de manière discrète sur la grille de pixels $(k-1,j-1)$, $1\le k\le n$, $0\le j\le m$.

(1) Déterminer, comme à la question 3 de l'exercice 2, la taille d'une matrice rectangulaire `II` telle que toute transformée par rotation autour du point $((k-1)/2,(j-1)/2)$ de l'image `I` conserve son support dans le cadre de cette matrice. À quel point du plan $\mathbb{R}^2$ correspond le pixel numéroté comme $(1,1)$ dans cette matrice `II` ?

(2) Après avoir initié la matrice `II` en une matrice nulle, modifiez ses entrées pour qu'elle devienne l'image de la transformée de l'image `I` par rotation d'angle θ autour de son centre $((k-1)/2,(l-1)/2)$ (complétée si nécessaire par des zéros). Vous utiliserez ici le procédé d'interpolation correspondant à la métode bilinéaire. Vous concevrez le code

```
>> [Iinit,Irot] = rotationdirect(I,theta);
```

de manière à ce qu'il retourne deux images de même taille :
- l'image `I` bordée de zéros de manière symétrique, à la fois horizontalement et verticalement (`zeropadding`), matrice notée `Iinit` ;
- la transformée par rotation de cette image, notée `Irot`, représentée dans le même cadre que celui dans lequel est représentée l'image `I` (devenue maintenant, une fois convenablement bordée de zéros, l'image `Iinit`).

EXERCICE 4 (une maquette de CAT-scanner virtuel). En exploitant les éléments de la routine `rotationdirect` élaborée à l'exercice 3, réalisez une routine `CATscanner`

```
>> S = CATscanner(I,tau);
```

qui, étant donnée une image digitale `I` :

- la place dans un premier temps dans un cadre suffisamment grand pour que l'on puisse l'y faire tourner d'un angle arbitraire $\theta \in [-\pi, \pi]$;
- calcule, pour les valeurs de $\theta = 0 : \tau : \pi$, le vecteur ligne réalisé en additionnant les lignes de l'image `I` pivotée de θ ;
- renvoie le sinogramme `S`, c'est-à-dire une image dont les entrées-lignes sont précisément les vecteurs lignes successifs obtenus pour les diverses valeurs de θ variant de 0 à π avec le pas τ.

L'image `S` ainsi construite est ce que l'on appelle le *sinogramme* de l'image `I`. C'est cette image que permet d'obtenir, coupe verticale par coupe verticale, la caméra en rotation dans le dispositif de CAT-scanner. Expliquez pourquoi. Vous pouvez tester cette routine sur les images artificielles générées par :

```
>> I = phantom(N);
```

(≪ fantôme de Shepp-Logan ≫) ou sur les images `CATscan1` ou `CATscan2` en ligne sur mon site.