subpixel_denis_duval_devoir5

November 24, 2022

1 Sub-Pixel Devoir 5

2 Exercice 17

3 Question 2

Avec une interpolation linéaire, le signal v est donné par : $v(2k) = u(k)$ et $v(2k-1) = \frac{u(k)+u(k-1)}{2}$. Sa transformée de Fourier vaut donc $\hat{v}(\xi) = \sum_{k\in\mathbb{Z}} u(k)e^{-i2k\xi} + \frac{e^{i\xi}}{2}\sum_{k\in\mathbb{Z}} u(k)e^{-i2k\xi} + \frac{e^{i\xi}}{2}e^{-i2\xi}\sum_{k\in\mathbb{Z}} u(k-1)e^{-i(k-1)2\xi}$. On reconnait la transformée de Fourier de u pris en 2ξ:

$$\hat{v}(\xi) = \hat{u}(2\xi)(1+\cos(\xi))$$

Avec l'interpolation de Shannon, on a $w(k) = U(k/2)$ ou U est l'interpolée de Shannon de u sur $\mathbb{R}$. On a $U = u * sinc$. Calculons la transformée de Fourier du signal $k \to U(k/2)$:

$$\hat{w}(\xi) = \sum_{k\in\mathbb{Z}}\sum_{m\in\mathbb{Z}} u(m)sinc(k/2-m)e^{-ik\xi}$$

$$\hat{w}(\xi) = \sum_{m\in\mathbb{Z}} u(m)\sum_{k\in\mathbb{Z}} sinc((k-2m)/2)e^{-ik\xi}$$

$$\hat{w}(\xi) = \sum_{m\in\mathbb{Z}} u(m)e^{-im2\xi}\sum_{k\in\mathbb{Z}} sinc((k-2m)/2)e^{-i(k-2m)\xi}$$

$$\hat{w}(\xi) = \hat{u}(2\xi)\hat{t}(\xi)$$

ou $t(k) = sinc(k/2)$. De plus on peut verifier facilement que $t(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-ik\xi/2}d\xi$ pour tout k. En faisant le changement de variable $u = \xi/2$:

$$t(k) = \frac{1}{2\pi}\int_{-\pi/2}^{\pi/2} 2e^{-ik\xi}d\xi = \frac{1}{2\pi}\int_{-\pi}^{\pi} 2\cdot 1_{[-\pi/2,\pi/2]}(\xi)e^{-ik\xi}d\xi$$

. En reconnaissant la transformée de Fourier inverse, on a trouvé la transformée de Fourier de t:

$$\hat{w}(\xi) = \hat{u}(2\xi)\cdot 2\cdot 1_{[-\pi/2,\pi/2]}(\xi)$$

(formule valable sur $[-\pi,\pi]$, on peut l'étendre à tout $\mathbb{R}$ en périodisant l'indicatrice).

On remarque que pour les deux interpolations on a une compression spectrale puisque les transformées de Fourier en ξ dépendent de celle de u en 2ξ. C'est-à-dire que le spectre sur $[-\pi,\pi]$ est compressé sur $[-\pi/2, \pi/2]$ avec un facteur 2 constant pour l'interpolation de Shannon et un facteur décroissant de 2 a 1 pour l'interpolation linéaire. Cependant, pour l'interpolation linéaire, on a également un ajout de fréquence car les fréquences initialement sur $[-2\pi, 2\pi]$ (non dans $[-\pi, \pi]$) sont compressées sur le spectre $[-\pi, \pi]$ avec une atténuation de 1 a 0. Pour l'interpolation de Shannon on n'a pas d'ajout de fréquence en raison de l'indicatrice.

4 Question 3

v est une image définie sur $0, ..., 2M-1$ x $0, ..., 2N-1$. Avec une interpolation bilinéaire, on a : $v(2k, 2l) = u(k, l)$, $v(2k+1, 2l) = \frac{u(k,l)+u(k+1,l)}{2}$, $v(2k, 2l+1) = \frac{u(k,l)+u(k,l+1)}{2}$ et $v(2k+1, 2l+1) = \frac{u(k,l)+u(k+1,l)+u(k,l+1)+u(k+1,l+1)}{4}$. On a pour la transformée de Fourier:

$$\hat{v}(q, r) = \sum_{k=0}^{2M-1} \sum_{l=0}^{2N-1} v(k, l) e^{-2i\pi(kq/2M+lr/2N)}$$

$$\hat{v}(q, r) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k, l) e^{-2i\pi(2kq/2M+2lr/2N)} + \frac{1}{2} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k, l) e^{-2i\pi((2k+1)q/2M+2lr/2N)} + \frac{1}{2} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k+1, l) e^{-2i\pi(}$$

On reconnait la transformée de Fourier de u en faisant des changements de variables et en utilisant la condition de bord périodique:

$$\hat{v}(q, r) = \hat{u}(q, r) + \frac{1}{2}\hat{u}(q, r)e^{-i\pi q/M} + \frac{1}{2}\hat{u}(q, r)e^{i\pi q/M} + \frac{1}{2}\hat{u}(q, r)e^{-i\pi r/N} + \frac{1}{2}\hat{u}(q, r)e^{i\pi r/N} + + \frac{1}{4}\hat{u}(q, r)e^{-i\pi q/M}e^{-i\pi r/N} + \frac{1}{4}$$

$$\hat{v}(q, r) = \hat{u}(q, r)(1 + \cos(\pi q/M))(1 + \cos(\pi r/N))$$

.

Pour l'interpolation de Shannon discrete, on observe que w est la convolution en 2D de u et de sinus cardinaux discrets. Comme les sinus cardinaux sont respectivement M et N périodiques, la transformée de Fourier de w est le produit des transformées de Fourier. On peut identifier la transformée de Fourier du signal $k \rightarrow sincd_M(k + x)$ avec x fixe grâce à la formule : $sincd_M(y) = Re(1/M \sum_{a=-M/2}^{M/2-1} e^{2i\pi ya/M})$. On en déduit, avec un calcul similaire au calcul précédent:

$$\hat{w}(q, r) = \hat{u}(q, r)(1 + e^{i\pi q/M})(1 + e^{i\pi r/N})$$

5 Exercice 18

[ ]:
```
import imtools as im
import numpy as np

bouc, cameraman = im.load('crop_bouc.pgm').astype('double'), im.
 ↪load('crop_cameraman.pgm').astype('double')
ns = [0, 1, 3, 5, -3]
"""for n in ns:
```

```
  u = im.fzoom(bouc, 16, n)
  v = im.fzoom(cameraman, 16, n)
  im.View(v)
"""

bouc, cameraman = im.load('bouc.pgm').astype('double'), im.load('cameraman.
 ↪pgm').astype('double')
fb = np.fft.fft2(bouc)
fc = np.fft.fft2(cameraman)
im.View(im.normsat(np.fft.fftshift(np.abs(fb)),1))
im.View(im.normsat(np.fft.fftshift(np.abs(fc)),1))
```

On voit que la meilleure méthode d'interpolation pour l'image crop_bouc semble être l'interpolation spline d'ordre 3 ou 5, tandis que la meilleure pour crop_cameraman semble être la bicubique.