AIC

Cours TC5

Result of TP4

Students:
Xiaoxiao CHEN
Dong FEI

Supervisor:
Prof. Yohanne TENDERO

13 oct. 2016

1 Exercise: Gaussian Filter using TFD

This exercise is implemented in the form of a function:

$$function\ image\_G = gaussianFiltre(I, sigma)$$

, and the Gaussian Filter is implemented in the frequency domain.

1.1 Input

$$zoom\_factor = 2;$$
$$sigma = 0.8 \times \sqrt{(zoom\_factor^2 - 1)}$$

We choose the picture with colorful houses as the input. Here the size of the picture is (512,512,3) which has 512×512 pixels and 3 color channels. The parameter $zoom_factor$ gives the scale of zooming out, and sigma is the Gaussian standard deviation corresponding to this zoom factor.

Figure 1: The origin picture

1.2 Gaussian Filter

The one-dimensional Gaussian filter has an impulse response expressed with the standard deviation σ as parameter

$$g(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

and the frequency response is given by

$$\hat{g}(f) = e^{-\frac{f^2}{2\sigma_f^2}}$$

With two dimensions, it is the product of two such Gaussians, one per direction:

$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and the frequency response is given by

$$\hat{g}(f_1, f_2) = e^{-2\pi^2\sigma^2(f_1^2+f_2^2)}$$

In Gaussian filter, the focal element receives the heaviest weight (having the highest Gaussian value) and neighboring elements receive smaller weights as their distance to the focal element increases. In Image processing, each element in the matrix represents a pixel attribute such as a color intensity, and the overall effect is called **Gaussian blur**.

The result of the Gaussian Filter:

Figure 2: Gaussian Filter

2 Exercise: Zoom out using under-sampling

This exercise is implemented with two functions.

$$function\ I^{'} = octave(I, zoom\_factor, nstep)$$

in which the zero padding function (in TP2) is implemented to maintain the origin size:

$$function\ I'' = zeroPadding(I_{'}, zoom\_factor)$$

For the zooming-out treatment, the main steps is as follows:

1) Gaussian Filter with the

$$standard\ deviation = 0.8 \times \sqrt{(zoom\_factor^2 - 1)}$$

the result of the Gaussian Filter as the section above;

2) Under-Sampling;

3) Compare the performance of 2) with or without 1).

2.1 Comparison of the result

To better compare the result of under-sampling based on Gaussian-blur and direct under-sampling, a post-processing was implemented to latter result. Zero-padding was used to enlarge the image to its original size so that we can compare them under an intuitive way.

It is noted that it is possible to call the *otave* function with the parameter $nstep = n > 1$. In this way, the under-sampling is operated n times. Here in this discussion, we use $nstep = 1$.

Figure 3: Under-sampling with Gaussian Filter

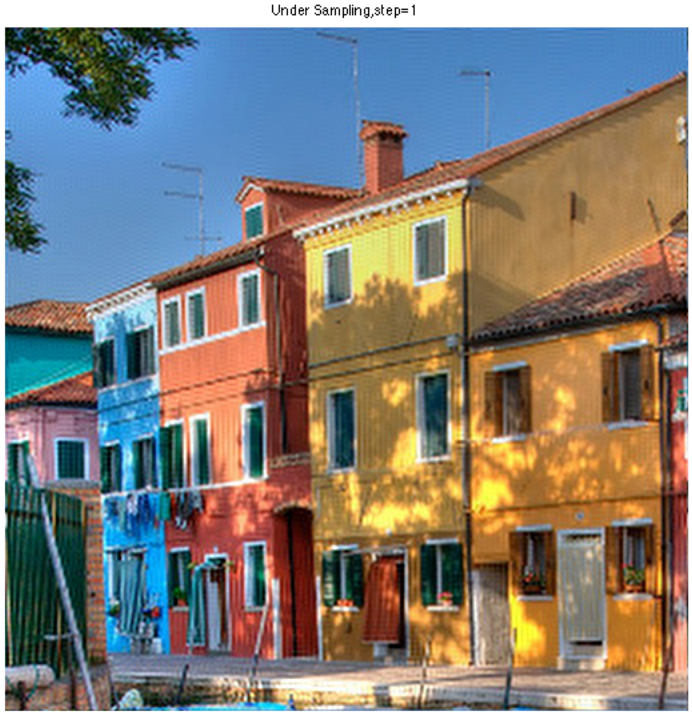

Figure 4: Under-sampling without Gaussian Filter

From the results, we can tell clearly that the under-sampling has the effect of aliasing. Using direct under-sampling, the strong visual effect of poor pixelizations can be observed,like different bricks piling up within the image. The antenna lost a little bit his connectivity, not smooth enough as its original figure. The contour is able to be detected but the detail information is hard to tell. The clothes outside the windows are totally blur, we shall try very hard to find the characteristics of these objects.

On the other hand, before going through under-sampling, images preprocessed with Gaussian Filter can effectually reduces the unpleasant effects caused by aliasing. The pixelizations is waken compared to direct undersampling, although the image seems more blur than before, we still feel its within connectivity and the contours of any objects can be detected. This experiment shows the potential uses of Gaussian Filter to neutralize the noise in the image.

3 Exercise: Laplacian algorithm

3.1 Laplacian operator

The Laplacian is often used as a image sharpening technique:

$$\Delta u(x,y) = \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y)$$

To calculate the Laplacian, the second derivatives of the two directions are calculated as follows:

$$\begin{aligned}
\frac{\partial u}{\partial x}(x,y) &= u(x,y) - u(x+1,y) \\
\frac{\partial u}{\partial y}(x,y) &= u(x,y) - u(x,y+1) \\
\frac{\partial^2 u}{\partial x^2}(x,y) &= \frac{\partial u}{\partial x}(x,y) - \frac{\partial u}{\partial x}(x+1,y) \\
\frac{\partial^2 u}{\partial y^2}(x,y) &= \frac{\partial u}{\partial y}(x,y) - \frac{\partial u}{\partial y}(x,y+1)
\end{aligned}$$

3.2 The Laplacian algorithm

This exercise is implemented with the function:

$$function\ I = Laplacien(I, eps, iter)$$

where *eps* is parameter which decides the performance of sharpening, the larger, the picture cleaner; *iter* is the number of loops.

The image sharpening is implemented with several iterations of Laplacian. A side effect of the Laplacian image sharpening is that, after each iteration, the picture will lose the last two rows and columns of pixels:

$$u(M,N) \to u(M-2, N-2)$$

That is to say, if $iteration = 10$, a certain part of the picture will be cut.

3.3 Comparison of the result

To further analyse the performance of Laplacian, sevral parameters are sent to the function, namely, a series of epsilons=[0.05,0.1,0.2], and iteration=[5,10,15]. Below are the cross-compared results.

(a) eps=0.05,iteration=5 (b) eps=0.05,iteration=10 (c) eps=0.05,iteration=15

(d) eps=0.1,iteration=5 (e) eps=0.1,iteration=10 (f) eps=0.1,iteration=15

(g) eps=0.2,iteration=5 (h) eps=0.2,iteration=10 (i) eps=0.2,iteration=15

Figure 5: Comparison of sharpening results

It is obvious that

- As the epsilon increase, the picture is cleaner, with edge enhanced;

However if *epsilon* is too large, 0.3 for example, the edges will be too bright, at the risk of loosing information.

- As the iteration increase, the picture is also sharpened;However, distortion also appears when iteration is too large. Moreover,a certain part of the picture is cut as iteration grows.

To conclude, the pair of *epsilon* and *iteration* effective enough but not too large is recommended.

4 Exercise: Z-transformation

4.1

$H(z) = 1 - \frac{1}{2}Z^{-1}$
$h(n) = \delta(n) - \frac{1}{2}\delta(n-1)$
$y_n = X_n = \frac{1}{2}X_{n-1}$
$\forall n < 0, h_n = 0$
$\Rightarrow causal$

4.2

$H(z) = 1 + 2Z^{-1} + 3Z^{-2}$
$h(n) = \delta(n) + 2\delta(n-1) + 3\delta(n-2)$
$y_n = X_n = X_n + 2X_{n-1} + 3X_{n-2}$
$\forall n < 0, h_n = 0$
$\Rightarrow causal$

4.3

$H(z) = \frac{1}{1-\frac{1}{2}Z^{-1}}$
$h(n) = \begin{cases} -(\frac{1}{2})^n U[n-1] & \text{if } |z| < \frac{1}{2} \Rightarrow anti-causal \\ (\frac{1}{2})^n U[n] & \text{if } |z| \geq \frac{1}{2} \Rightarrow causal \end{cases}$
$y_n = X_n = X_n + \frac{1}{2}X_{n-1} + \frac{1}{4}X_{n-2} + ... + (\frac{1}{2})^k X_{n-k}$

4.4

$H(z) = \frac{1}{1-2Z^{-1}}$

$h(n) = \begin{cases} -2^n U[-n-1] & \text{if } |z| < 2 \Rightarrow anti - causal \\ 2^n U[n] & \text{if } |z| \geq 2 \Rightarrow causal \end{cases}$

$y_n = X_n = X_n + 2X_{n-1} + 4X_{n-2} + ... + 2^k X_{n-k}$