DIP Homework Assignment #4

Name: 方郁婷

ID #: r10922196

email: r10922196@ntu.edu.tw

Problem 1: DIGITAL HALFTONING

(a) (10 pt) According to the dither matrix I_2, please perform dithering to obtain a binary image result1.png.

```
I2 = np.array([[1, 2], [3, 0]], dtype = 'uint8')
N = len(I2)
threshold = 255 * (I2 + 0.5) / (N * N)
threshold
```

```
array([[ 95.625, 159.375],
       [223.125,  31.875]])
```

```
h, w = sample1.shape
threshold = np.tile(threshold, (h//N, w//N))
threshold
```

```
array([[ 95.625, 159.375,  95.625, ..., 159.375,  95.625, 159.375],
       [223.125,  31.875, 223.125, ...,  31.875, 223.125,  31.875],
       [ 95.625, 159.375,  95.625, ..., 159.375,  95.625, 159.375],
       ...,
       [223.125,  31.875, 223.125, ...,  31.875, 223.125,  31.875],
       [ 95.625, 159.375,  95.625, ..., 159.375,  95.625, 159.375],
       [223.125,  31.875, 223.125, ...,  31.875, 223.125,  31.875]])
```

```
result1 = np.zeros((h,w))
result1[sample1 >= threshold] = 1
```

將 I2 擴大成與圖片一樣大後，判斷有無過 Threshold，有過的話就設為 1

	
input	output

看起來相當粗糙

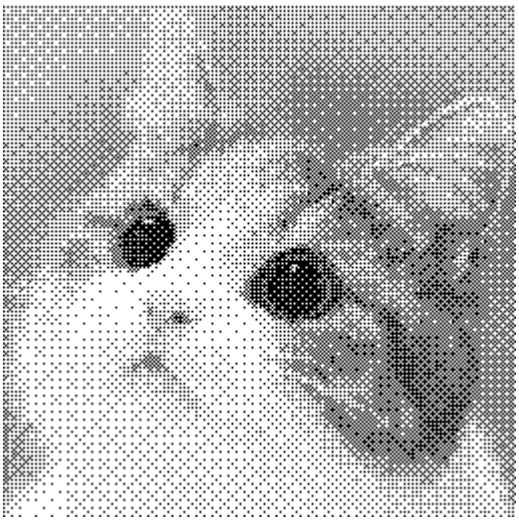
(b) (15 pt) Expand the dither matrix I_2 to I_{256} (256 × 256) and use it to perform dithering. Output the result as result2.png. Compare result1.png and result2.png along with some discussions.

$$I_{2n}(i,j)=\begin{bmatrix} 4I_n(i,j)+1 & 4I_n(i,j)+2 \\ 4I_n(i,j)+3 & 4I_n(i,j)+0 \end{bmatrix}$$

```
n = 2
I = np.array([[1, 2], [3, 0]], dtype = 'uint8')

for i in range(7):
    I2 = np.zeros((n*2, n*2))

    I2[0:n, 0:n] = I*4 + 1
    I2[0:n, n: ] = I*4 + 2
    I2[n: , 0:n] = I*4 + 3
    I2[n: , n: ] = I*4 + 0

    I = I2
    n*=2

I256 = I
```

將原矩陣擴充到(256X256)後再使用跟上一題一樣的方法

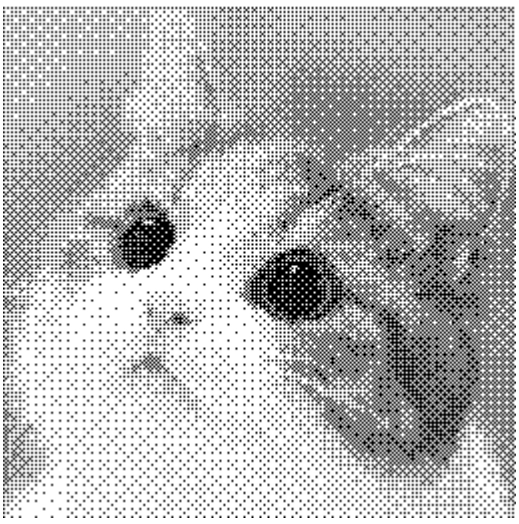
input



output

有變比較精緻一點

Result1

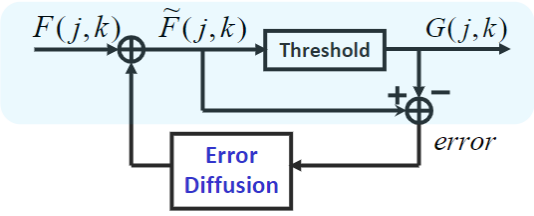


Result2

Result1 看起來還是會有四四方方的網格樣

Result2 的網格有較多變化

(c) (25 pt) Perform error diffusion with Floyd-Steinberg and Jarvis' patterns on sample1.png. Out- put the results as result3.png and result4.png, respectively. You may also try more patterns and show the results in your report. Discuss these patterns based on the results.

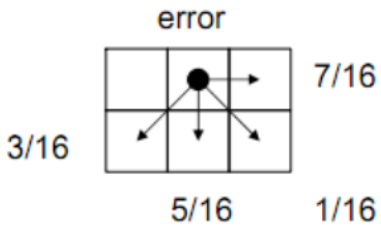


$$if \quad \tilde{F}(j,k) \geq 0.5 \quad \rightarrow \quad G(j,k) = 1$$

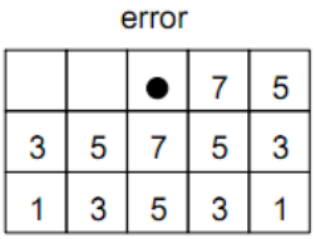
$$if \quad \tilde{F}(j,k) < 0.5 \quad \rightarrow \quad G(j,k) = 0$$

$$Define \quad E(j,k) = \tilde{F}(j,k) - G(j,k)$$

- **1975 Floyd Steinberg:**



1976 Jarvis et al:



(1/48)

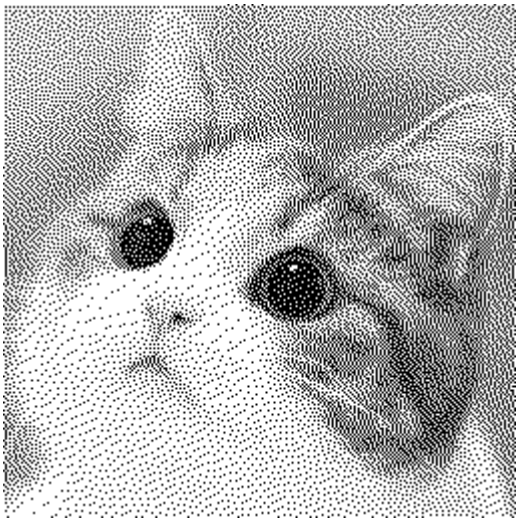
```
# Floyd Steinberg
result3 = np.copy(np.lib.pad(sample1,(1,1),'constant')) / 255

kernel = [[0, 0, 7/16],
        [3/16, 5/16, 1/16]]

ones = np.ones((2,3))

Height, Width = result3.shape

for y in range(1, Height-1):
    for x in range(1, Width-1):

        old_value = result3[y, x]
        new_value = 0
        if (old_value >= 0.5) :
            new_value = 1

        Error = old_value - new_value

        patch = result3[y:y+2, x-1:x+2]

        NewNumber = patch + Error * ones * kernel
        NewNumber[NewNumber>1] = 1
        NewNumber[NewNumber<0] = 0

        result3[y:y+2, x-1:x+2] = NewNumber
        result3[y, x] = new_value

result3 = result3[1:257, 1:257]
```

以 Floyd-Steinberg 為例

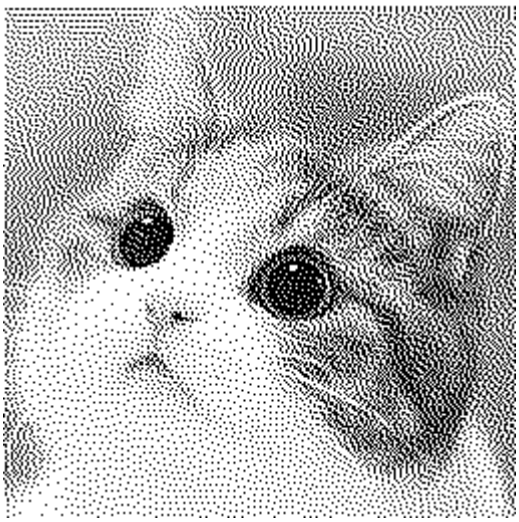
首先先判斷有無過 Threshold 給出 new_value，之後計算 Error，再將 Error 擴散給周邊 pixel(使用 kernel 計算)，擴散完後再將超過 1 的設為 1，小於 0 的設為 0

Jarvis'的算法同上，只是改變 kernel 矩陣裡的值

input



Floyd-Steinberg



Jarvis’

Discuss:

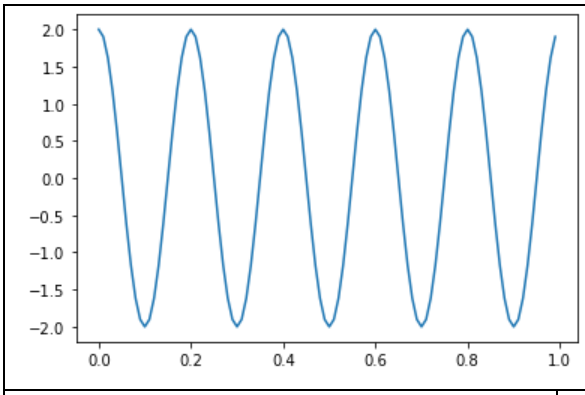
Jarvis’的眼白還有右邊鬍鬚比較明顯，空格與對比較大，Floyd-Steinberg 比較暗比較糊。

Problem 2: Image Sampling

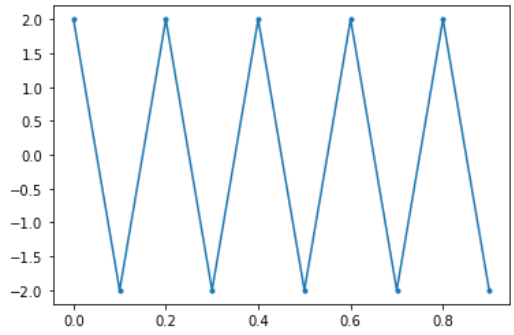
(a) (25 pt) By analyzing sample2.png, please explain how to perform image sampling on it to avoid aliasing. Please also perform ’inappropriate’ image sampling which results in aliasing in the sampled image. Output the result as result5.png, specify the sampling rate you choose and discuss how it affects the resultant image.

老實說我在上課時一直半聽半懂的，因為那些公式看起來真的好可怕，不是很好理解，所以我就重新學了一下一維的 sample

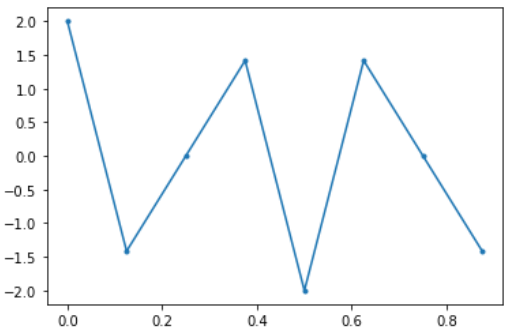
以一維的波為例



原波

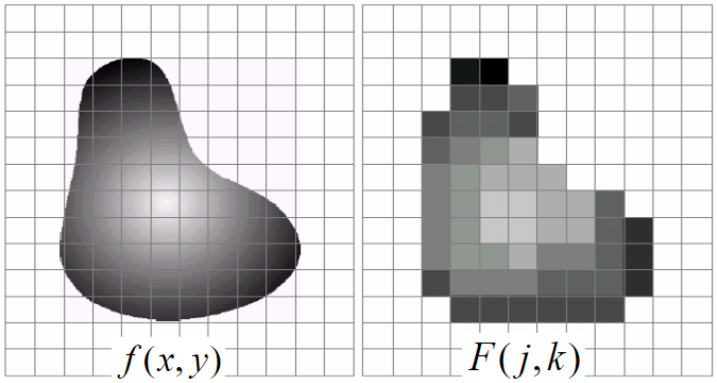


剛好不失真
Sample rate = 2 倍頻率



aliasing
Sample rate < 2 倍頻率

之後又突然想到這跟老師之前在課堂上講過的圖片 sampling 很像，就一瞬間茅塞頓開了。

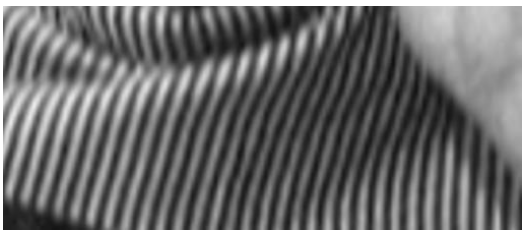


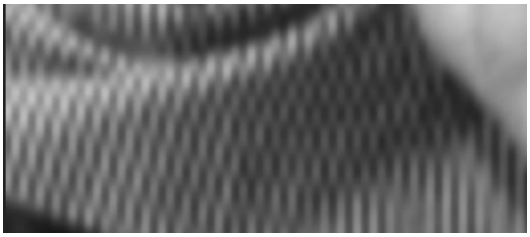
在空間域上可以把 sample 的頻率視為圖片解析度，sample 後跟原圖比有很明顯的 aliasing，我想在頻率域上也是一樣的原理吧。

以二維的例子來說

```
ratio = 0.5
sample_image = cv2.resize(sample2, dsize=[0,0], fx=ratio, fy=ratio)
```

當你的 ratio 設為 0.5 時，你其實只保留了二分之一的 col 與 row，因此你的圖片喪失了一半的資訊，所以會有 aliasing，如同一維的例子一樣



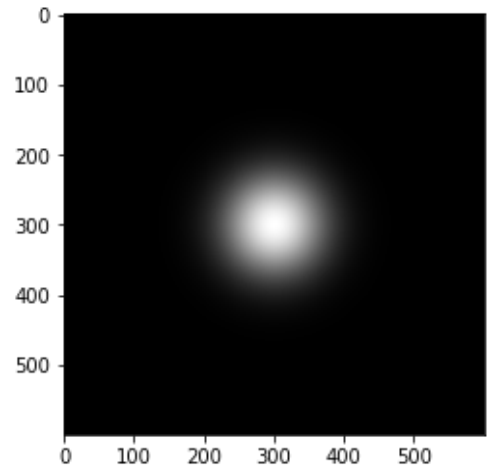
input | output

Sample 後的條紋變網格了

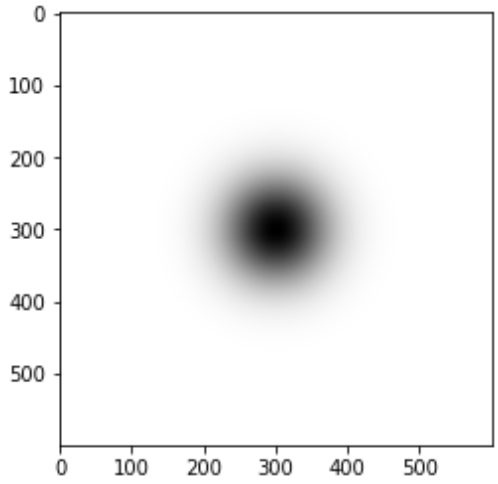
以這張圖來說，ratio 設 0.8 以下就會產生明顯的 aliasing，要避免產生 aliasing 就要讓 Sample rate >= 2 倍頻率

(b) (25 pt) Given sample3.png, please perform the unsharp masking mentioned in the lecture in the frequency domain and transform the result back to the pixel domain by inverse Fourier transform.
Save the resultant image as result6.png and describe your steps in detail.

我使用的是高斯的 mask，因為比較不會有水波紋



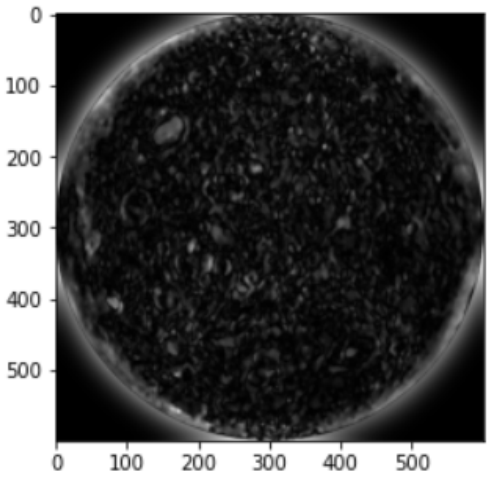
這是 low pass 的 mask

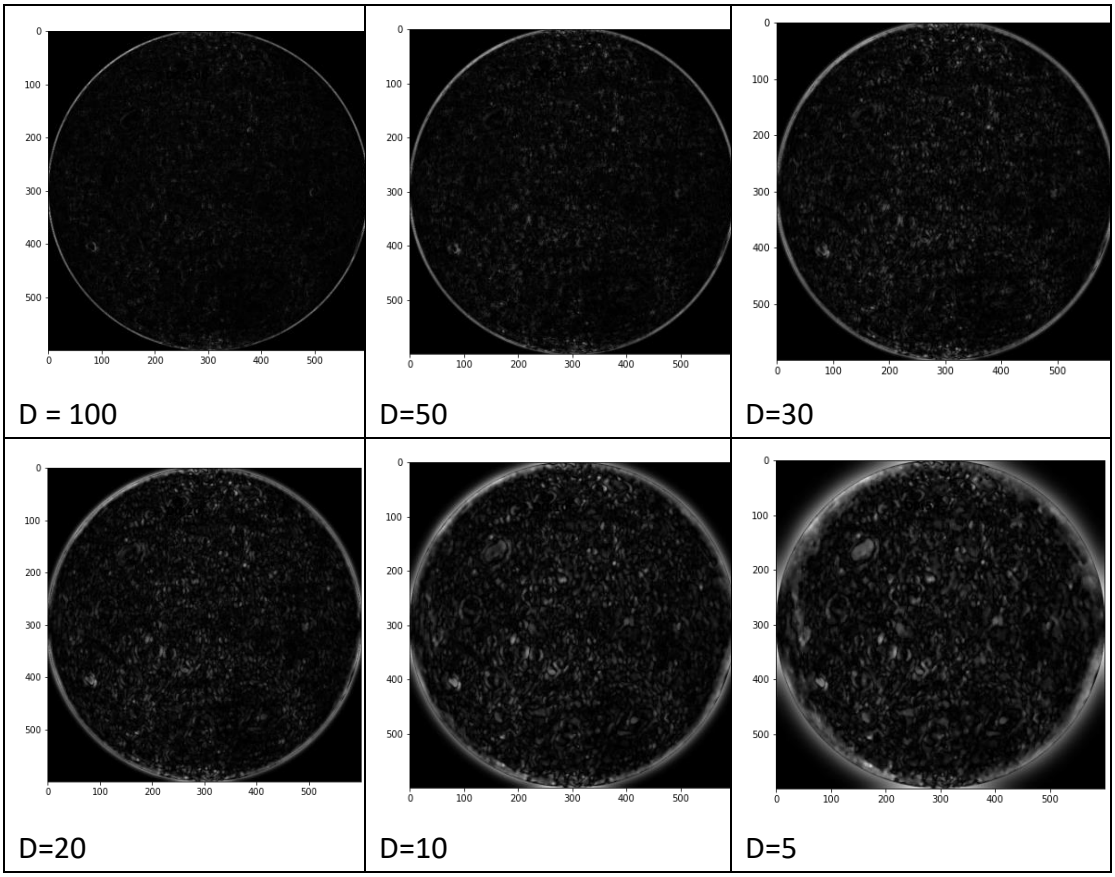


這是 1 - low pass 的 mask = high pass

```
d0 = 5
original = np.fft.fft2(img) #傅立葉轉換
center = np.fft.fftshift(original) #將座標(0,0)轉到中心
HighPassCenter = center * gaussianHP(d0,img.shape) #乘以 high pass mask
HighPass = np.fft.ifftshift(HighPassCenter) #將座標轉回去
inverse_HighPass = np.fft.ifft2(HighPass) #做逆傅立葉轉換
result6 = np.abs(inverse_HighPass)
result6 = (result6-result6.min())/(result6.max()-result6.min())*255
plt.imshow(result6, "gray")
```

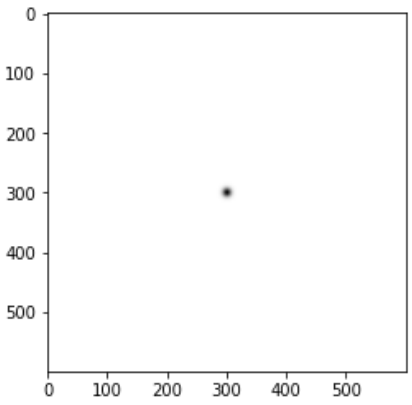
```
<matplotlib.image.AxesImage at 0x2521dbed700>
```

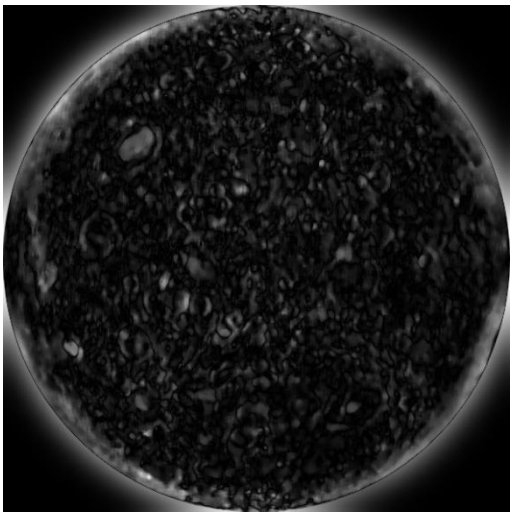




D 越小，月球表面越明顯

我最後選擇 D=5 的 mask(如下圖)，因為他能把月球的凹凸輪廓變得比較明顯



	
input	output