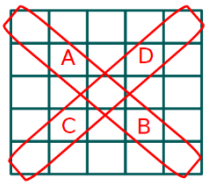
數位影像處理：Kawahera 濾波器實作

01157026 馮宥崴

實習內容：

1. 了解 Kawahera 濾波器的原理
2. 使用以下鄰近區域定義實作 Kawahera 濾波器



實習目標：

- 熟悉 Kawahera 濾波器的操作

實習步驟：

1. 詳讀 Kawahera 濾波器的程式碼，並了解其原理

```python
from skimage import io
import scipy.ndimage as ndi
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import convolve2d

def Kuwahara(original, winsize):
    image = original.astype(np.float64)
    # make sure window size is correct
    if winsize %4 != 1:
        raise Exception ("Invalid winsize %s: winsize must follow formula: w = 4*n+1." %winsize)

    #Build subwindows
    tmpAvgKerRow = np.hstack((np.ones((1,(winsize-1)//2+1)),np.zeros((1,(winsize-1)//2))))
    tmpPadder = np.zeros((1,winsize))
    tmpavgker = np.tile(tmpAvgKerRow, ((winsize-1)//2+1,1))
    tmpavgker = np.vstack((tmpavgker, np.tile(tmpPadder, ((winsize-1)//2,1))))
    tmpavgker = tmpavgker/np.sum(tmpavgker)

    # tmpavgker is a 'north-west' subwindow (marked as 'a' above)
    # we build a vector of convolution kernels for computing average and
    # variance
    avgker = np.empty((4,winsize,winsize)) # make an empty vector of arrays
    avgker[0] = tmpavgker           # North-west (a)
    avgker[1] = np.fliplr(tmpavgker)    # North-east (b)
    avgker[2] = np.flipud(tmpavgker)    # South-west (c)
    avgker[3] = np.fliplr(avgker[2])    # South-east (d)
    # Create a pixel-by-pixel square of the image
    squaredImg = image**2
# preallocate these arrays to make it apparently %15 faster
    avgs = np.zeros([4, image.shape[0],image.shape[1]])
    stddevs = avgs.copy()

    # Calculation of averages and variances on subwindows
    for k in range(4):
        # mean on subwindow
        avgs[k] = convolve2d(image, avgker[k],mode='same')
        # mean of squares on subwindow
        stddevs[k] = convolve2d(squaredImg, avgker[k],mode='same')
        # variance on subwindow
        stddevs[k] = stddevs[k]-avgs[k]**2
    # Choice of index with minimum variance
    indices = np.argmin(stddevs,0) # returns index of subwindow with smallest variance

    # Building the filtered image (with nested for loops)
    filtered = np.zeros(original.shape)
    for row in range(original.shape[0]):
        for col in range(original.shape[1]):
            filtered[row,col] = avgs[indices[row,col], row,col]

    #filtered=filtered.astype(np.uint8)
    return filtered.astype(np.uint8)
```

2. 透過閱讀程式碼，我們可以瞭解到：

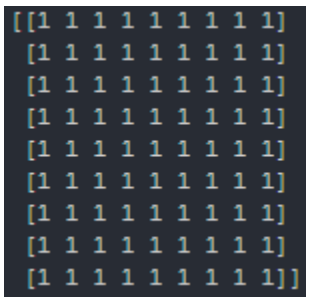
```
#Build subwindows
tmpAvgKerRow = np.hstack((np.ones((1,(winsize-1)//2+1)),np.zeros((1,(winsize-1)//2))))
tmpPadder = np.zeros((1,winsize))
tmpavgker = np.tile(tmpAvgKerRow, ((winsize-1)//2+1,1))
tmpavgker = np.vstack((tmpavgker, np.tile(tmpPadder, ((winsize-1)//2,1))))
tmpavgker = tmpavgker/np.sum(tmpavgker)

# tmpavgker is a 'north-west' subwindow (marked as 'a' above)
# we build a vector of convolution kernels for computing average and
# variance
avgker = np.empty((4,winsize,winsize)) # make an empty vector of arrays
avgker[0] = tmpavgker           # North-west (a)
avgker[1] = np.fliplr(tmpavgker)    # North-east (b)
avgker[2] = np.flipud(tmpavgker)    # South-west (c)
avgker[3] = np.fliplr(avgker[2])    # South-east (d)
```

這一段程式用於定義鄰近區域的圖形，我們只需要修程式碼即可

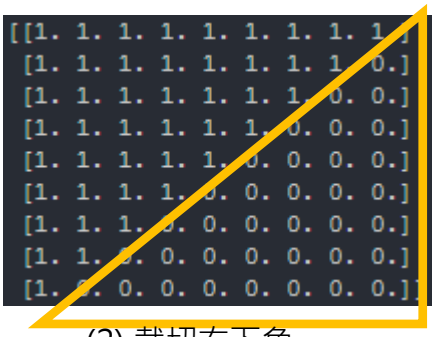
3. 定義鄰近區域的圖形

```
tmpavgker = np.triu(np.full((winsize, winsize), 1) - np.rot90(np.tri(winsize, winsize, -1)))
tmpavgker = tmpavgker / np.sum(tmpavgker)
# tmpavgker is a 'north-west' subwindow (marked as 'a' above)
# we build a vector of convolution kernels for computing average and
# variance
avgker = np.empty((4,winsize,winsize)) # make an empty vector of arrays
avgker[0] = tmpavgker
avgker[1] = np.rot90(avgker[0])
avgker[2] = np.rot90(avgker[1])
avgker[3] = np.rot90(avgker[2])
print(avgker)
```

步驟：

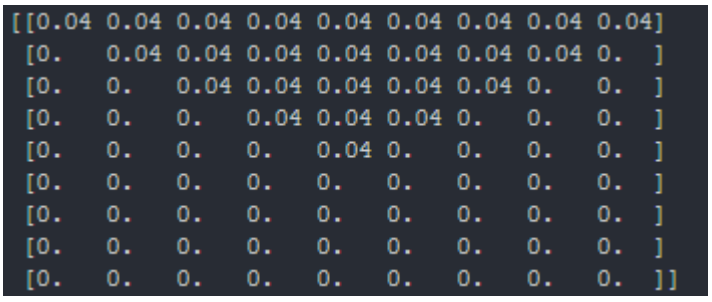


(1) 生成二維矩陣



(2) 裁切右下角

(3) 過濾掉左下角



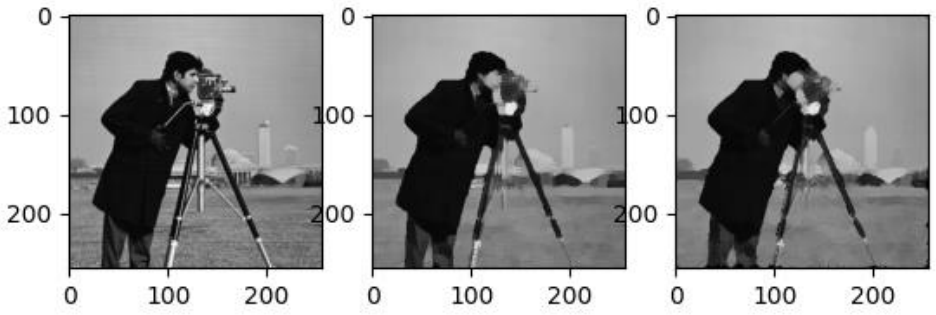
(4) 賦值

程式碼：

```
from skimage import io
import scipy.ndimage as ndi
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import convolve2d

def Kuwahara(original, winsize):
    image = original.astype(np.float64)
    # make sure window size is correct
    if winsize %4 != 1:
        raise Exception ("Invalid winsize %s: winsize must follow formula: w = 4*n+1." %winsize)

    #Build subwindows
    tmpAvgKerRow = np.hstack((np.ones((1,(winsize-1)//2+1)),np.zeros((1,(winsize-1)//2))))
    tmpPadder = np.zeros((1,winsize))
    tmpavgker = np.tile(tmpAvgKerRow, ((winsize-1)//2+1,1))
    tmpavgker = np.vstack((tmpavgker, np.tile(tmpPadder, ((winsize-1)//2,1))))
    tmpavgker = tmpavgker/np.sum(tmpavgker)

    # tmpavgker is a 'north-west' subwindow (marked as 'a' above)
    # we build a vector of convolution kernels for computing average and
    # variance
    avgker = np.empty((4,winsize,winsize)) # make an empty vector of arrays
    avgker[0] = tmpavgker           # North-west (a)
    avgker[1] = np.fliplr(tmpavgker)    # North-east (b)
    avgker[2] = np.flipud(tmpavgker)    # South-west (c)
    avgker[3] = np.fliplr(avgker[2])    # South-east (d)
    # Create a pixel-by-pixel square of the image
    squaredImg = image**2
# preallocate these arrays to make it apparently %15 faster
    avgs = np.zeros([4, image.shape[0],image.shape[1]])
    stddevs = avgs.copy()

    # Calculation of averages and variances on subwindows
    for k in range(4):
        # mean on subwindow
        avgs[k] = convolve2d(image, avgker[k],mode='same')
        # mean of squares on subwindow
        stddevs[k] = convolve2d(squaredImg, avgker[k],mode='same')
        # variance on subwindow
        stddevs[k] = stddevs[k]-avgs[k]**2
    # Choice of index with minimum variance
    indices = np.argmin(stddevs,0) # returns index of subwindow with smallest variance

    # Building the filtered image (with nested for loops)
    filtered = np.zeros(original.shape)
    for row in range(original.shape[0]):
        for col in range(original.shape[1]):
            filtered[row,col] = avgs[indices[row,col], row,col]

    #filtered=filtered.astype(np.uint8)
    return filtered.astype(np.uint8)

def KuwaharaDiag(original, winsize):
    image = original.astype(np.float64)
    # make sure window size is correct
    if winsize %4 != 1:
        raise Exception ("Invalid winsize %s: winsize must follow formula: w = 4*n+1." %winsize)

    tmpavgker = np.triu(np.full((winsize, winsize), 1) - np.rot90(np.tri(winsize, winsize, -1)))
    tmpavgker = tmpavgker / np.sum(tmpavgker)
    # tmpavgker is a 'north-west' subwindow (marked as 'a' above)
    # we build a vector of convolution kernels for computing average and
    # variance
    avgker = np.empty((4,winsize,winsize)) # make an empty vector of arrays
    avgker[0] = tmpavgker
    avgker[1] = np.rot90(avgker[0])
    avgker[2] = np.rot90(avgker[1])
    avgker[3] = np.rot90(avgker[2])
    print(avgker)
    # Create a pixel-by-pixel square of the image
    squaredImg = image**2
# preallocate these arrays to make it apparently %15 faster
    avgs = np.zeros([4, image.shape[0],image.shape[1]])
    stddevs = avgs.copy()

    # Calculation of averages and variances on subwindows
    for k in range(4):
        # mean on subwindow
        avgs[k] = convolve2d(image, avgker[k],mode='same')
        # mean of squares on subwindow
```

```
        stddevs[k] = convolve2d(squaredImg, avgker[k],mode='same')
        # variance on subwindow
        stddevs[k] = stddevs[k]-avgs[k]**2
    # Choice of index with minimum variance
    indices = np.argmin(stddevs,0) # returns index of subwindow with smallest variance

    # Building the filtered image (with nested for loops)
    filtered = np.zeros(original.shape)
    for row in range(original.shape[0]):
        for col in range(original.shape[1]):
            filtered[row,col] = avgs[indices[row,col], row,col]

    #filtered=filtered.astype(np.uint8)
    return filtered.astype(np.uint8)

c = io.imread('Digital-Image-Processing\\Lab5\\cameraman.tif')

cK=Kuwahara(c,9)
cKD=KuwaharaDiag(c, 9)



fig = plt.figure()
plt.gray()  # show the filtered result in grayscale
ax1 = fig.add_subplot(131)  # left side
ax2 = fig.add_subplot(132)  # right side
ax3 = fig.add_subplot(133)

ax1.imshow(c/255,vmax=1.0,vmin=0.0)
ax2.imshow(cK/255,vmax=1.0,vmin=0.0)
ax3.imshow(cKD/255, vmax=1.0, vmin=0.0)

plt.show()
```

實習結果：



左：原圖；中：Kawahera 濾波；右：Kawahera 對角濾波（實習結果）

實習心得：

這次的實習相對較為簡單，不需要過多複雜的程式，但就是要熟悉 Numpy 的用法，否則程式碼會變得較為冗長。