國立中與大學

一一學年度第一學期

數位影像處理

第三次平時作業

班級: 資工碩士一年級

學號:7110056210

學生:丁吾心

授課教師:吳俊霖 教授

目錄

適應性中值濾波器 Adaptive median filter.	2
1.1 題目說明	2
1.2 整體流程	
1.3詳細流程與說明	4
1.3.1 整體流程	4
1.3.2 中值濾波器	4
1.3.3 適應性中值濾波器	5
1.4 結果	6
1.5 程式碼	9
16 心得分享	11

適應性中值濾波器 Adaptive median filter

1.1 題目說明

這次作業是在影像上撒上胡椒鹽雜訊,使用中值濾波器將雜訊給 濾掉,分別使用適應性中值濾波器以及一般中值濾波器。

適應性中值濾波器演算法如下

LevelA:
$$A1 = z_{med} - z_{min}$$
$$A2 = z_{med} - z_{max}$$

If
$$A1 > 0$$
 AND $A2 < 0$, Go to level B

Else increase the window size

If window size $\leq S_{\text{max}}$ repeat level A

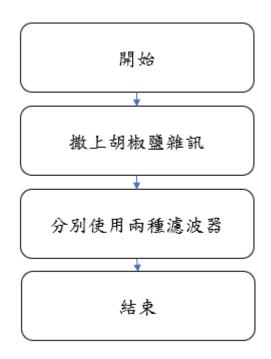
Else output z_{xy}

LevelB:
$$B1 = z_{xy} - z_{\min}$$

$$B2 = z_{xy} - z_{\max}$$

If B1 > 0 AND B2 < 0, output z_{xy} Else output z_{med} e/se

1.2 整體流程



因 1.1 有說明演算法,這邊不再贅述。

1.3 詳細流程與說明

1.3.1 整體流程

```
def main():
   test_name = 'test.jpg'
   gray img = cv2.imread( test name, cv2.IMREAD GRAYSCALE )
   noise_img = gray_img.copy()
   print( gray img.shape )
   h, w = gray_img.shape
   noise num = int( 0.5 * h * w )
   for index in range( noise_num ):
       x = random.randint(0, h - 1)
       y = random.randint(0, w - 1)
       if random.randint( 0, 1 ) == 0:
           noise img[x, y] = 0
       else:
           noise img[x, y] = 255
   a img = noise img.copy()
   m img = noise img.copy()
   final M img = Median filter( m img )
                                                  # 中值濾波器
   final_A_img = Adaptive_Median_filter( a_img ) # 適應性中個
```

1.3.2 中值濾波器

取排序濾波器中的中間值,設為影像值

```
def Median_filter( img ):
    ret_img = img.copy()
    ret_img = cv2.copyMakeBorder( ret_img, 3, 3, 3, 3, cv2.BORDER_CONSTANT, value = 0 )
    for i in range( 3, img.shape[ 0 ] - 3 ):
        for j in range( 3, img.shape[ 1 ] - 3 ):
            window_img = img[ i-1:i+2, j-1:j+2 ]
            window_img = window_img.reshape( 1, -1 )
            s = np.sort( window_img[ 0 ] )  # 排序
            ret_img[ i, j ] = s[ 4 ]  # 取中間值
    return ret_img[ 3:img.shape[ 0 ] - 3, 3:img.shape[ 1 ] - 3]
```

1.3.3 適應性中值濾波器

```
def Adaptive_Median_filter( img ):
    ret_img = img.copy()
    ret_img = cv2.copyMakeBorder( ret_img, 3, 3, 3, cv2.BORDER_CONSTANT, value = 0 )
    window_size = [ 3, 5, 7 ]
    for i in range( 3, img.shape[ 0 ] - 3 ):
        for j in range( 3 , img.shape[ 1 ] - 3 ):
             for w in window_size:
                                                                   # 在 Window 中跑
                ret_img[ i, j ] = img[ i, j ]
f = math.floor( w / 2 )
                 c = math.ceil( w / 2 )
                 window_img = img[ i-f:i+c, j-f:j+c ]
                                                                   # Window 範圍
                 window_img = window_img.reshape( 1, -1 )
s = np.sort( window_img[ 0 ] )
                 m = math.floor( ( w * w ) / 2 )
                 min = s[0]
med = s[m]
max = s[-1]
                                                                   # 最大
                 if med <= min or med >= max:
                      if img[ i, j ] <= min or img[ i, j ] >= max:
    ret_img[ i, j ] = med
    return ret_img[ 3:img.shape[ 0 ] - 3, 3:img.shape[ 1 ] - 3]
```

1.4 結果



原圖



0.25 雜訊圖

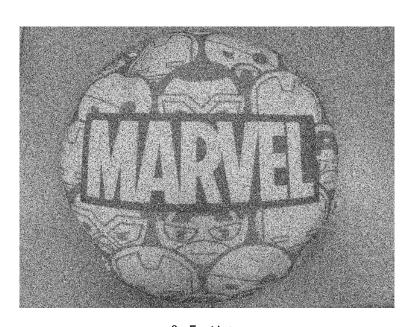


中值濾波器



適應性中值濾波器

小結:可以看到在雜訊 0.25 的情況下,適應性中值濾波器比中值濾波器的效果略高一些,但中值濾波器的效果勉強可以,接下來是 0.5 的雜訊



0.5 雜訊



中值濾波器



適應性中值濾波器

在 0.5 的雜訊中就很明顯,適應性中值濾波器的效果要好上許多。

1.5 程式碼

```
import numpy as np
import random
import math
import cv2
def Adaptive_Median_filter( img ):
   ret_img = img.copy()
   ret_img = cv2.copyMakeBorder( ret_img, 3, 3, 3, cv2.BORDER_CONSTANT,
value = 0)
   window_size = [3, 5, 7]
                                                            # 3*3 5*5 7*7
的 Window Size
   for i in range(3, img. shape[0] - 3):
       for j in range(3, img. shape[1]-3):
           for w in window_size:
                                                           # 在 Window 中
跑
               ret_img[i, j] = img[i, j]
               f = math. floor( w / 2 )
               c = math.ceil(w / 2)
               window_img = img[i-f:i+c, j-f:j+c]
                                                          # Window 範圍
               window_img = window_img.reshape( 1, -1 )
               s = np. sort( window_img[ 0 ] )
                                                          #排序
               m = math.floor((w * w) / 2)
                                                          # 最小
               min = s[0]
                                                          #中間
               med = s[m]
               \max = s[-1]
                                                          # 最大
               if med <= min or med >= max:
                   continue
                                                           # 若不是 min <
med < max 則去下一個 Window
               else:
                   if img[i, j] \le min \text{ or } img[i, j] \ge max:
                       ret_img[ i, j ] = med
                   break
   return ret_img[ 3:img.shape[ 0 ] - 3, 3:img.shape[ 1 ] - 3]
def Median_filter( img ):
   ret_img = img.copy()
```

```
ret_img = cv2.copyMakeBorder( ret_img, 3, 3, 3, cv2.BORDER_CONSTANT,
value = 0
   for i in range(3, img. shape[0] - 3):
       for j in range(3, img. shape[1]-3):
           window_img = img[i-1:i+2, j-1:j+2]
           window img = window img.reshape(1, -1)
                                                      #排序
           s = np. sort( window_img[ 0 ] )
           ret_img[i, j] = s[4]
                                                      #取中間值
   return ret_img[ 3:img. shape[ 0 ] - 3, 3:img. shape[ 1 ] - 3]
def main():
   test_name = 'test.jpg'
   gray_img = cv2. imread( test_name, cv2. IMREAD_GRAYSCALE )
   noise_img = gray_img.copy()
   print( gray_img. shape )
   h, w = gray_img. shape
   noise_num = int( 0.5 * h * w )
   for index in range( noise_num ):
       x = random. randint(0, h - 1)
       y = random. randint(0, w - 1)
       if random. randint(0, 1) == 0:
           noise_img[x, y] = 0
       else:
           noise_img[x, y] = 255
   a_img = noise_img.copy()
   m_img = noise_img.copy()
   final M img = Median filter( m img )
                                          # 中值濾波器
   final_A_img = Adaptive_Median_filter(a_img) # 適應性中值濾波器
   cv2. imshow('Gray', gray_img')
   cv2. imshow( 'Noise', noise_img )
   cv2. imshow('Median', final_M_img')
   cv2.imshow('Adaptive', final_A_img)
   cv2. imwrite( 'Gray. jpg', gray_img )
   cv2. imwrite('Noise. jpg', noise_img')
```

```
cv2.imwrite( 'Median.jpg', final_M_img )
cv2.imwrite( 'Adaptiven.jpg', final_A_img )
cv2.waitKey()
cv2.destroyAllWindows()
```

main()

1.6 心得分享

我覺得適應性中值濾波器的效果比中值濾波器好很多,不僅可以保留住本來的細節,也可以把雜訊都濾掉,因中值濾波器在 window 的大小無法調整,所以可能在雜訊密度高的部分會處理得不好。