

原圖：



640 * 640



1280 * 960



4032 * 3024

主函式：

```
167 if __name__ == '__main__':
168     image = cv2.imread('img/Block.jpeg') # 讀取要處理的檔案
169     sign = cv2.imread('img/sign.jpeg') # 讀取要用的檔案
170     print("1:resize_Bilinear_interpolation\n" +
171           "2:resize_Bilinear_interpolation\n" +
172           "3:filter_Average_blur\n" +
173           "4:filter_Median_blur\n" +
174           "5:filter_Gaussian_blur\n")
175     key_controls = {'1': NN_interpolation, # 建立函式索引的dic
176                    '2': Bilinear_interpolation,
177                    '3': Average_blur,
178                    '4': Median_blur,
179                    '5': Gaussian_blur}
180     a = input("choose:")
181     # b = input("put sign(y/n):")
182     key_controls[a](image, sign) # 呼叫所選的函式
```

可選擇要什麼功能，選擇後可選擇要變成的尺寸大小，或者是 filter 的大小。

放大縮小圖片：

- 最近鄰：(700 * 1000)

```
6 def NN_interpolation(image, sign): # 最近鄰內插 (縮小放大圖片)
7     h = int(input("h(高度):"))
8     w = int(input("w(寬度):"))
9     rows, cols, _c = image.shape # (640, 640, 3)
10     create = np.zeros((h, w, 3), np.uint8) # 建立一個所要縮圖的影像矩陣
11     for i in range(h):
12         for j in range(w):
13             a = round(i*(cols/w)) # 取出目標矩陣對應於原本矩陣的座標 (a,b) = (x,y)
14             b = round(j*(rows/h))
15             create[i, j] = image[b, a] # 把原本矩陣的像素值給目標矩陣
16     # 加上簽名
17     rows, cols, _c = create.shape # (640, 640, 3)
18     resize_sign = cv2.resize(sign, dsize=(100, 50))
19     create[(rows-50):(rows), (cols-100):(cols)] = resize_sign
20
21     h = str(h)
22     w = str(w)
23     create = create.astype(np.uint8) # 把目標矩陣轉換成特定的dtype，不然有機會跑錯
24     cv2.imshow("NN_interpolation_range:"+"h"+h+"w"+w, create)
25     # cv2.imwrite("img/NN_interpolation_"+"h"+h+"w"+w+".jpeg", create)
26     cv2.waitKey(0)
27
```



黃怡偉



黃怡偉



黃怡偉

- 線性：(700 * 1000)

```

19 def Bilinear_interpolation(image, sign): # 雙線性 (插小放大圖片)
20     h = int(input("%s高度:" % sign))
21     w = int(input("%s寬度:" % sign))
22     rows, cols, c = image.shape
23     create = np.zeros((h, w, 3), np.uint8) # 建立一個待插值的圖像矩陣
24     for y in range(h):
25         for x in range(w):
26             # 取出插值區域的左上角和右下角的座標 (float_x = x + 0.5, float_y = y + 0.5)
27             # 取中心點座標 (0.5座標的部份)
28             float_x = round(x + (cols - w) / 2, 1)
29             float_y = round(y + (rows - h) / 2, 1)
30             # 分別取出四個鄰近座標
31             u, l = math.floor(float_x)
32             v, j = math.floor(float_y)
33             i = int(i)
34             j = int(j)
35             if (i <= (cols - 1) or j <= (rows - 1)): # 當所選擇座標在邊緣時，讓它取用於原本矩陣的邊緣的座標值
36                 create[y, x] = image[j, i]
37             else: # a1, a2, a3, a4 分別為對邊緣座標最近的四個像素
38                 a1 = image[j, i]
39                 a2 = image[j, i+1]
40                 a3 = image[j+1, i]
41                 a4 = image[j+1, i+1]
42             # vals = (1-u)*(1-v)*a1+u*(1-v)*a2+u*(1-v)*a3+v*(1-v)*a4 # 第三次內插公式 (先垂直再水平) , 求出四次內插值
43             vals = (1-u)*(1-v)*a1+u*(1-v)*a2+u*(1-v)*a3+v*(1-v)*a4 # 第三次內插公式 (先水平再垂直) , 求出四次內插值
44             create[y, x] = vals
45     # 加上顏色
46     rows, cols, c = create.shape # (640, 480, 3)
47     resize_sign = cv2.resize(sign, (w, h))
48     create[rows-h:(rows), cols-w:(cols)] = resize_sign
49     h = str(h)
50     w = str(w)
51     create = create.astype(np.uint8) # 把插值矩陣轉換成特定的type , 不然有警告訊息
52     cv2.imshow("Bilinear_interpolation_range", create)
53     #cv2.imwrite("img/Bilinear_interpolation_"+h+"*"+w+".jpg", create)
54     cv2.waitKey(0)

```



黃柏偉



黃柏偉



黃柏偉

- 比較不同方向的線性：



黃柏偉

先垂直再水平



黃柏偉

先水平再垂直

我感覺不出差異。

模糊圖片：(filter: (5 * 5))

- Average blur: (模糊最明顯，而且雜訊能被消除的應該不多)

```
70 def Average_blur(image, sign): # 均值濾波器
71     r = int(input("filter range:"))
72     rows, cols, _c = image.shape # (640,640,3)
73     create = np.zeros((rows, cols, 3), np.uint8) # 建立一個與要模糊的圖像同等大小的影像矩陣
74
75     for y in range(rows):
76         for x in range(cols):
77             vals = average_mask(r, image, x, y) # 得到通過平均濾波器(average_mask)的值
78             # print(vals)
79             create[y, x] = vals
80
81     # 加上簽名
82     rows, cols, _c = create.shape # (640,640,3)
83     resize_sign = cv2.resize(sign, dsize=(100, 50))
84     create[(rows-50):(rows), (cols-100):(cols)] = resize_sign
85
86     r = str(r)
87     create = create.astype(np.uint8) # 把目標矩陣轉成特定的dtype，不然有機會跑錯
88     cv2.imshow("Average_blur_"+r+"filter:"+r, create)
89     #cv2.imwrite("img/Average_blur_"+r+"filter:"+r+".jpeg", create)
90     cv2.waitKey(0)
```

```
92 def average_mask(r, image, x, y): # 平均濾波器遮罩
93     average = 0
94     rows, cols, _c = image.shape # (640,640,3)
95     u = int(r/2) # 所選遮罩的大小對應的座標範圍 (x+s,y+t) (-u <= (s,t) <= u)
96     for i in range(-u, u+1):
97         for j in range(-u, u+1):
98             if (x+j) > 0 and (x+j) < cols: # 當遮罩座標沒超過原本矩陣的範圍時
99                 if (y+i) > 0 and (y+i) < rows:
100                     # 計算每一点的平均像素值並累加
101                     average = average + image[(y+i), (x+j)]*(1/(r*r))
102     return average
```



- Median blur: (雖然看得出模糊，但沒有 Average blur 明顯)

```
105 def Median_blur(image, sign): # 中值濾波器
106     r = int(input("filter range:"))
107     rows, cols, _c = image.shape # (640,640,3)
108
109     create = np.zeros((rows, cols, 3), np.uint8) # 建立一個與要模糊的圖像同等大小的影像矩陣
110
111     for y in range(rows):
112         for x in range(cols):
113             vals = median_mask(r, image, x, y) # 得到通過中值濾波器(median_mask)的值
114             # print(vals)
115             create[y, x] = vals
116
117     # 加上簽名
118     rows, cols, _c = create.shape # (640,640,3)
119     resize_sign = cv2.resize(sign, dsize=(100, 50))
120     create[(rows-50):(rows), (cols-100):(cols)] = resize_sign
121
122     r = str(r)
123     create = create.astype(np.uint8) # 把目標矩陣轉成特定的dtype，不然有機會跑錯
124     cv2.imshow("Median_blur_"+r+"filter:"+r, create)
125     #cv2.imwrite("img/Median_blur_"+r+"filter:"+r+".jpeg", create)
126     cv2.waitKey(0)
```

```
128 def median_mask(r, image, x, y): # 中值濾波器遮罩
129     # 存放三個通道的像素值
130     median_b = []
131     median_g = []
132     median_r = []
133
134     rows, cols, _c = image.shape # (640,640,3)
135     u = int(r/2) # 所選遮罩的大小對應的座標範圍 (x+s,y+t) (-u <= (s,t) <= u)
136     for i in range(-u, u+1):
137         for j in range(-u, u+1):
138             if (x+j) > 0 and (x+j) < cols: # 當遮罩座標沒超過原本矩陣的範圍時
139                 if (y+i) > 0 and (y+i) < rows:
140                     # 存放所選座標遮罩範圍內三個通道的像素值
141                     median_b.append(image[(y+i), (x+j)].tolist()[0])
142                     median_g.append(image[(y+i), (x+j)].tolist()[1])
143                     median_r.append(image[(y+i), (x+j)].tolist()[2])
144     # 返回三個通道的中位數像素值
145     vals = (np.median(median_b), np.median(median_g), np.median(median_r))
146     # print(type(np.median(median_b)))
147     return vals
```



- Gaussian blur: (對照片的模糊影響較小)

```

150 def Gaussian_blur(image, sign): # 高斯濾波器
151     r = int(input("filter range:"))
152     # create = np.zeros((rows, cols, 3), np.uint8)
153
154     create = cv2.GaussianBlur(image, (r, r), 0) # opencv 的高斯函式 (0是偏差值)
155
156     # 加上簽名
157     rows, cols, _c = create.shape # (640,640,3)
158     resize_sign = cv2.resize(sign, dsize=(100, 50))
159     create[(rows-50):(rows), (cols-100):(cols)] = resize_sign
160
161     r = str(r)
162     cv2.imshow("Gaussian_blur_"+r, create)
163     #cv2.imwrite("img/Gaussian_blur_"+r+".jpeg", create)
164     cv2.waitKey(0)

```



- 三種方式在人臉上的差異：



Average Blur



Median Blur



Gaussian Blur

模糊化：高斯 < 中位數 < 平均，但不確定雜訊哪個消去比較多。

簽名檔：

```

57     # 加上簽名
58     rows, cols, _c = create.shape # (640,640,3)
59     resize_sign = cv2.resize(sign, dsize=(100, 50))
60     create[(rows-50):(rows), (cols-100):(cols)] = resize_sign

```

心得討論：

因為要利用程式碼來做出數學的實作，所以對於公式的印象以及了解相對於課堂會更加深刻。而因為本來就會寫點 `opencv`，所以對於像素的處理部分本身就有一些了解，但透過這次也更加深了如何處理，尤其對於濾波器的概念更加了解，這個對於我的專題應該會提供一些幫助。