HW1

姓名：蔡育嘉 學號：D0641798

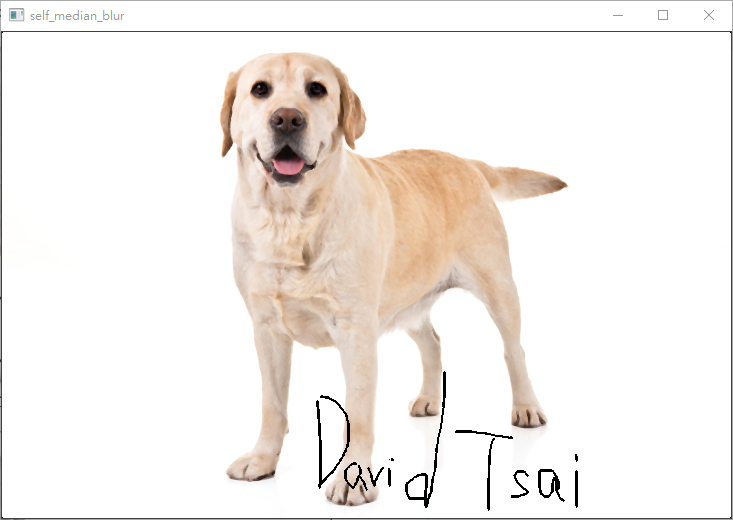
模糊圖片 (Average Blur、Medium Blur、Gaussian Blur)

# 一、程式碼：

|  |
| --- |
| import cv2 import numpy as np import skimage from skimage.viewer import ImageViewer import sys import math  img=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/cat.jpg")  cv2.imshow("Original",img)  print(img.shape)  filter\_size=3 #這邊可以改size extend\_size=int((filter\_size-1)/2)# 算要額外增加一圈的矩陣寬度是多少 # filter size= 3-> 1 增加一圈 # filter size= 5-> 2 增加二圈 以此類推  img\_len,img\_width,img\_channels=img.shape  y=np.zeros([img\_len+(extend\_size\*2),img\_width+(extend\_size\*2),img\_channels],dtype="uint8") #增加兩圈 陣列上下左右增加2倍  for channel in range(img\_channels):  for i in range(extend\_size,img\_len+extend\_size):  for j in range(extend\_size,img\_width+extend\_size): # 把圖片放進四周圍pixel值為0中  y[i][j][channel]=img[i-extend\_size][j-extend\_size][channel]   #卷積運算 def average\_blur(img,filter\_size):  img\_len,img\_width,img\_channels=img.shape  convolution=np.ones([filter\_size,filter\_size,img\_channels])  for channel in range(img\_channels):  for i in range(img\_len):  for j in range(img\_width):  temp=0  for filter\_X in range(filter\_size):  for filter\_Y in range(filter\_size):  temp=temp+y[i+filter\_X][j+filter\_Y][channel]\*convolution[filter\_X][filter\_Y][channel]  img[i][j][channel]=round(temp/(filter\_size\*filter\_size))   signature=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/signature\_cat.png")  signature\_x,signature\_y,signature\_channels=signature.shape  for channel in range(signature\_channels):  for i in range(signature\_x):  for j in range(signature\_y):  if signature[i][j][channel]==0:#如果簽名檔為黑 則self\_blur同位置也是黑的  img[i][j][channel]=0  cv2.imshow("self\_average\_blur",img)  cv2.waitKey()  #找中位數 def median\_blur(img,filter\_size):  img\_len,img\_width,img\_channels=img.shape  for channel in range(img\_channels):  for i in range(img\_len):  for j in range(img\_width):  temp=[]  for filter\_X in range(filter\_size):  for filter\_Y in range(filter\_size):  temp.append(y[i+filter\_X][j+filter\_Y][channel])  temp.sort() #把filter內的數字排列之後  half=int(len(temp)/2)#找index一半的元素  img[i][j][channel]=temp[half]   signature=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/signature\_cat.png")  signature\_x,signature\_y,signature\_channels=signature.shape  for channel in range(signature\_channels):  for i in range(signature\_x):  for j in range(signature\_y):  if signature[i][j][channel]==0:#如果簽名檔為黑 則self\_blur同位置也是黑的  img[i][j][channel]=0  cv2.imshow("self\_median\_blur",img)  cv2.waitKey()  def gaussian\_blur(img,filter\_size):   blur = cv2.GaussianBlur(img, (filter\_size, filter\_size),0)  signature=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/signature\_cat.png")  signature\_x,signature\_y,signature\_channels=signature.shape  for channel in range(signature\_channels):  for i in range(signature\_x):  for j in range(signature\_y):  if signature[i][j][channel]==0:#如果簽名檔為黑 則self\_blur同位置也是黑的  img[i][j][channel]=0  cv2.imshow("gaussian\_blur",img)  cv2.waitKey()    average\_blur(img,filter\_size) median\_blur(img,filter\_size) gaussian\_blur(img,filter\_size) |

# 二、執行結果：

# 



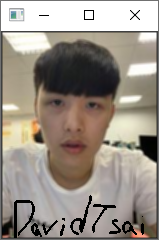








原圖



average\_blur median\_blur gaussian\_blur

**比較分析三種模糊方法在人臉照片下的差別**

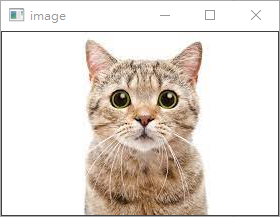
上面三種是用filter\_size=3的參數去執行，average\_blur的模糊程度最高，median\_blur跟gaussian\_blur的效果，肉眼看起來很像，最主要的差別我認為是像在臉頰的地方，median\_blur如果框到的地方都是皮膚色居多，陰影的地方就不會顯示出來，而gaussian\_blur在臉頰的部分，看起來多了陰影，沒有像medain\_blur的圖片來得紅潤。

# 縮放圖片(最近鄰、線性)

# 一、程式碼：

|  |
| --- |
| import numpy as np import cv2 from math import sqrt  import math  #img 原圖  #time 縮放倍數 def interpolation(img,time):  height,width,channels =img.shape  new\_height=int(height\*time)  new\_width=int(width\*time)   emptyImage=np.zeros((new\_height,new\_width,channels),dtype=np.uint8)  # -------  # | A B |  # | C D |  # -------  #  # -----------  # | A B |  # | |  # | |  # | C D |  # -----------  for channel in range(channels):  for y in range(new\_height):  for x in range(new\_width):    src\_X=x/time  src\_Y=y/time  if src\_X%1==0 and src\_Y%1==0: #如果XY是整數 就直接取元圖對應到的index  src\_X=int(src\_X)  src\_Y=int(src\_Y)  emptyImage[y][x][channel]=img[src\_Y][src\_X][channel]   else:  A\_x=int(src\_X)  A\_y=int(src\_Y)   B\_x=A\_x+1  B\_y=A\_y   C\_x=A\_x  C\_y=A\_y+1   D\_x=A\_x+1  D\_y=A\_y+1    if D\_y>=height: #D超過的時候 微調ABCD座標  D\_y=height-1  C\_y=height-1  A\_y=C\_y-1  B\_y=D\_y-1  if D\_x>=width:  D\_x=width-1  B\_x=width-1  A\_x=B\_x-1  C\_x=D\_x-1   A=img[A\_y][A\_x][channel]  B=img[B\_y][B\_x][channel]  C=img[C\_y][C\_x][channel]  D=img[D\_y][D\_x][channel]   A=int(A)  B=int(B)  C=int(C)  D=int(D)    w=x-(A\_x\*time)  W=(B\_x-A\_x)\*time  h=y-(A\_y\*time)  H=(C\_y-A\_y)\*time   i=A+(w\*(B-A)/W)  j=C+((w\*(D-C))/W)    pixel=i+((h\*(j-i))/H)  emptyImage[y][x][channel]=pixel    return emptyImage    def interpolation\_test(img,time):  height,width,channels =img.shape  new\_height=int(height\*time)  new\_width=int(width\*time)   emptyImage=np.zeros((new\_height,new\_width,channels),dtype=np.uint8)  # -------  # | A B |  # | C D |  # -------  #  # -----------  # | A B |  # | |  # | |  # | C D |  # -----------  for channel in range(channels):  for y in range(new\_height):  for x in range(new\_width):    src\_X=x/time  src\_Y=y/time  if src\_X%1==0 and src\_Y%1==0: #如果XY是整數 就直接取元圖對應到的index  src\_X=int(src\_X)  src\_Y=int(src\_Y)  emptyImage[y][x][channel]=img[src\_Y][src\_X][channel]   else:  A\_x=int(src\_X)  A\_y=int(src\_Y)   B\_x=A\_x+1  B\_y=A\_y   C\_x=A\_x  C\_y=A\_y+1   D\_x=A\_x+1  D\_y=A\_y+1    if D\_y>=height: #D超過的時候 微調ABCD座標  D\_y=height-1  C\_y=height-1  A\_y=C\_y-1  B\_y=D\_y-1  if D\_x>=width:  D\_x=width-1  B\_x=width-1  A\_x=B\_x-1  C\_x=D\_x-1   A=img[A\_y][A\_x][channel]  B=img[B\_y][B\_x][channel]  C=img[C\_y][C\_x][channel]  D=img[D\_y][D\_x][channel]   A=int(A)  B=int(B)  C=int(C)  D=int(D)    w=x-(A\_x\*time)  W=(B\_x-A\_x)\*time  h=y-(A\_y\*time)  H=(C\_y-A\_y)\*time  #先從y方向算 再算x  i=A+(h\*(C-A)/H)  j=B+((h\*(D-B))/H)    pixel=i+((w\*(j-i))/W)  emptyImage[y][x][channel]=pixel    return emptyImage def nearest(img,time):  height,width,channels =img.shape   new\_height=int(height\*time)  new\_width=int(width\*time)  # -------  # | A B |  # | C D |  # -------   # -----------  # | A B |  # | |  # | |  # | C D |  # -----------  emptyImage=np.zeros((new\_height,new\_width,channels),dtype=np.uint8)  for channel in range(channels):  for i in range(new\_height):  for j in range(new\_width):  y=int(i/time) #除放大或縮小的倍率  x=int(j/time) #取整數  emptyImage[i][j][channel]=img[y][x][channel]   return emptyImage   img=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/me.jpg")  print(img.shape) self\_interpolation=interpolation(img,2)  self\_nearest=nearest(img,2) self\_interpolation\_test=interpolation\_test(img,2)  signature=cv2.imread("C:/Users/user/Desktop/imageprocess/HW1/signature\_me.png")    signature=interpolation(signature,2) signature\_y,signature\_x,channels=signature.shape for channel in range(channels):  for y in range(signature\_y):  for x in range(signature\_x):  if signature[y][x][channel]==0:#如果簽名檔為黑 則nearest\_neighbor同位置也是黑的  self\_interpolation[y][x][channel]=0  self\_interpolation\_test[y][x][channel]=0  self\_nearest[y][x][channel]=0  self\_interpolation\_height,self\_interpolation\_width,channels=self\_interpolation.shape for channel in range(channels):  for y in range(self\_interpolation\_height):  for x in range(self\_interpolation\_width):  if self\_interpolation[y][x][channel]!=self\_interpolation\_test[y][x][channel]:  print(self\_interpolation[y][x][channel])  print(self\_interpolation\_test[y][x][channel])   print("OK") cv2.imshow("self\_interpolation",self\_interpolation) #cv2.imshow("self\_interpolation\_test",self\_interpolation\_test)      cv2.imshow("self\_nearest",self\_nearest) cv2.imshow("image",img) cv2.waitKey(0) |

# 二、執行結果：



放大兩倍

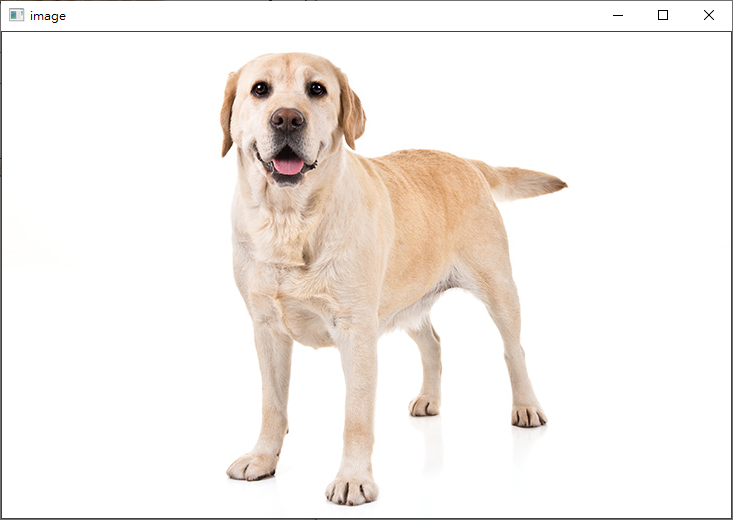




縮小2倍



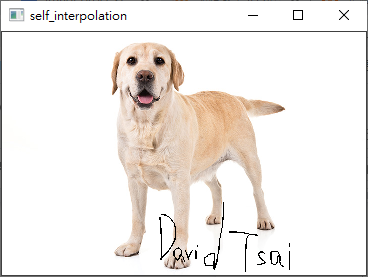
最近鄰 內插







這邊圖片太大了，我貼上來只能這樣，不過他整個視窗比例有變化





原圖

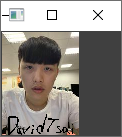
放大兩倍







縮小2倍



最近鄰 內插

**比較線性內插時從不同方向進行內插的結果**

我分別從兩個方向同時做內插，接著把內插結束的圖片，比較每個pixel出來的值是一樣的

