Pattern Recognition Computer Vision Project

學號：M0523816

姓名：黃羿衡

一、 實驗環境

- Python (3.6.4)
- numpy (1.14.2) : array 操作
- scikit-learn (0.19.1) : 正規化資料
- sklearn (0.0) : 正規化資料
- Pillow (5.0.0) : 圖片存取

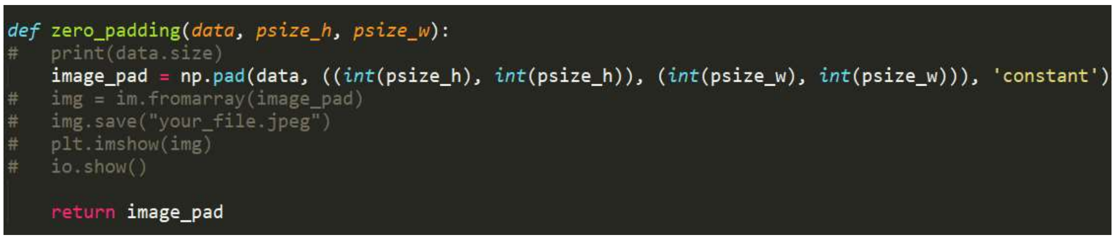
二、 程式碼:

題目操作寫在

if __name__ == '__main__':

Problem1

zero_padding(data, psize_h, psize_w):



```
def zero_padding(data, psize_h, psize_w):
#    print(data.size)
    image_pad = np.pad(data, ((int(psize_h), int(psize_h)), (int(psize_w), int(psize_w))), 'constant')
#    img = im.fromarray(image_pad)
#    img.save("your_file.jpeg")
#    plt.imshow(img)
#    io.show()

    return image_pad
```

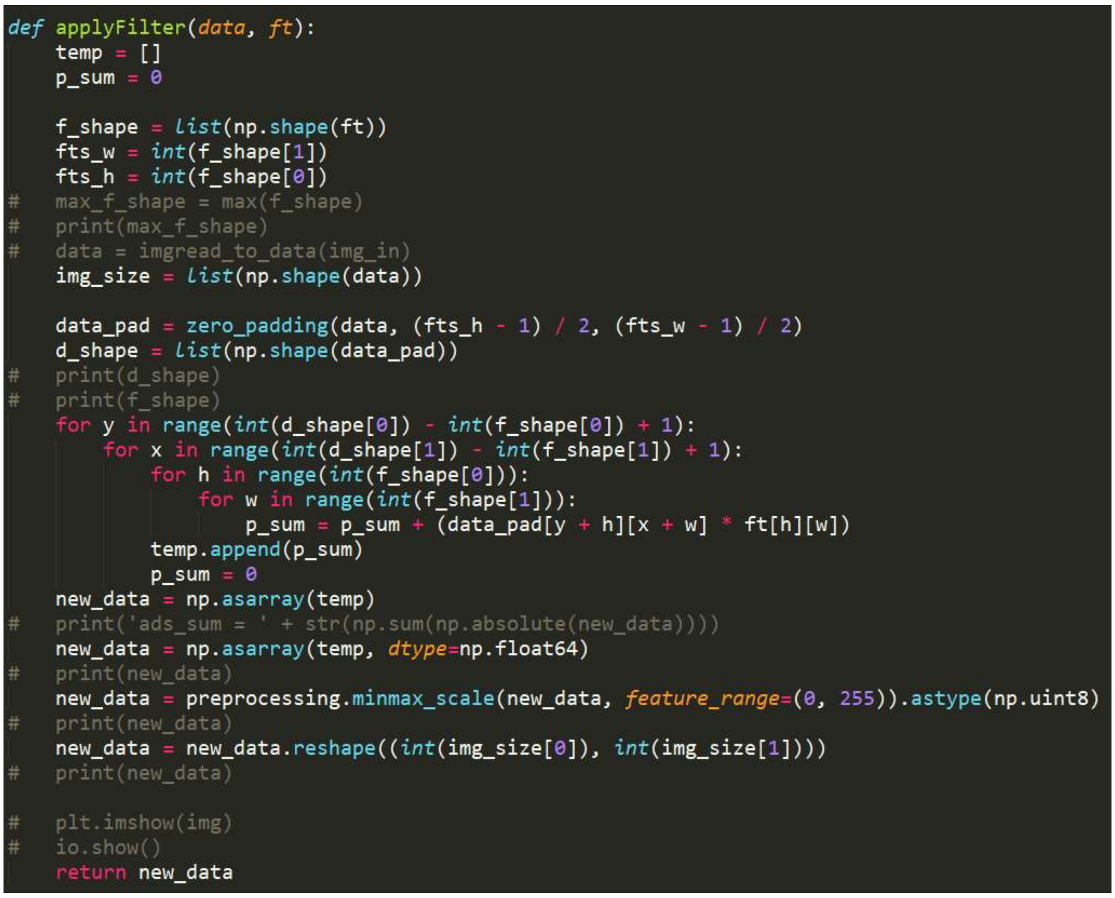
功能 :將圖片 padding 以防 filter 後圖片縮小

傳入 1 個 channel 的 image, 高的 padding size, 寬的 padding size

Padding 圖

Return padding 後的圖

applyFilter(data, ft):



```
def applyFilter(data, ft):
    temp = []
    p_sum = 0

    f_shape = list(np.shape(ft))
    fts_w = int(f_shape[1])
    fts_h = int(f_shape[0])
#   max_f_shape = max(f_shape)
#   print(max_f_shape)
#   data = imgread_to_data(img_in)
    img_size = list(np.shape(data))

    data_pad = zero_padding(data, (fts_h - 1) / 2, (fts_w - 1) / 2)
    d_shape = list(np.shape(data_pad))
#   print(d_shape)
#   print(f_shape)
    for y in range(int(d_shape[0]) - int(f_shape[0]) + 1):
        for x in range(int(d_shape[1]) - int(f_shape[1]) + 1):
            for h in range(int(f_shape[0])):
                for w in range(int(f_shape[1])):
                    p_sum = p_sum + (data_pad[y + h][x + w] * ft[h][w])
            temp.append(p_sum)
            p_sum = 0
    new_data = np.asarray(temp)
#   print('ads_sum = ' + str(np.sum(np.absolute(new_data))))
    new_data = np.asarray(temp, dtype=np.float64)
#   print(new_data)
    new_data = preprocessing.minmax_scale(new_data, feature_range=(0, 255)).astype(np.uint8)
#   print(new_data)
    new_data = new_data.reshape((int(img_size[0]), int(img_size[1])))
#   print(new_data)

#   plt.imshow(img)
#   io.show()
    return new_data
```

功能 ： 實作 filter

傳入 1 channel image, filter

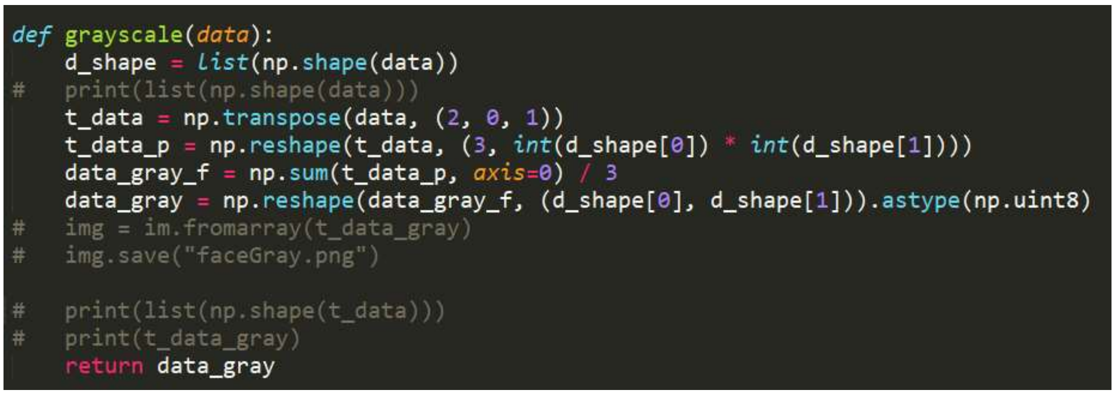
Padding 後開始做 filtering

將 filter 後的 data 轉型成 float64 和 flatten，為了後面將資料正規化到

0-255，再將結果 reshape 回原本高寬

Return filter 後所得到的圖

Problem2
grayscale(data):



```python
def grayscale(data):
    d_shape = list(np.shape(data))
#   print(list(np.shape(data)))
    t_data = np.transpose(data, (2, 0, 1))
    t_data_p = np.reshape(t_data, (3, int(d_shape[0]) * int(d_shape[1])))
    data_gray_f = np.sum(t_data_p, axis=0) / 3
    data_gray = np.reshape(data_gray_f, (d_shape[0], d_shape[1])).astype(np.uint8)
#   img = im.fromarray(t_data_gray)
#   img.save("faceGray.png")

#   print(list(np.shape(t_data)))
#   print(t_data_gray)
    return data_gray
```

功能 ： 將彩色圖片灰階化

傳入一張 color image

將圖片 shape (h, w, c) transpose to (c, h, w)方便使用

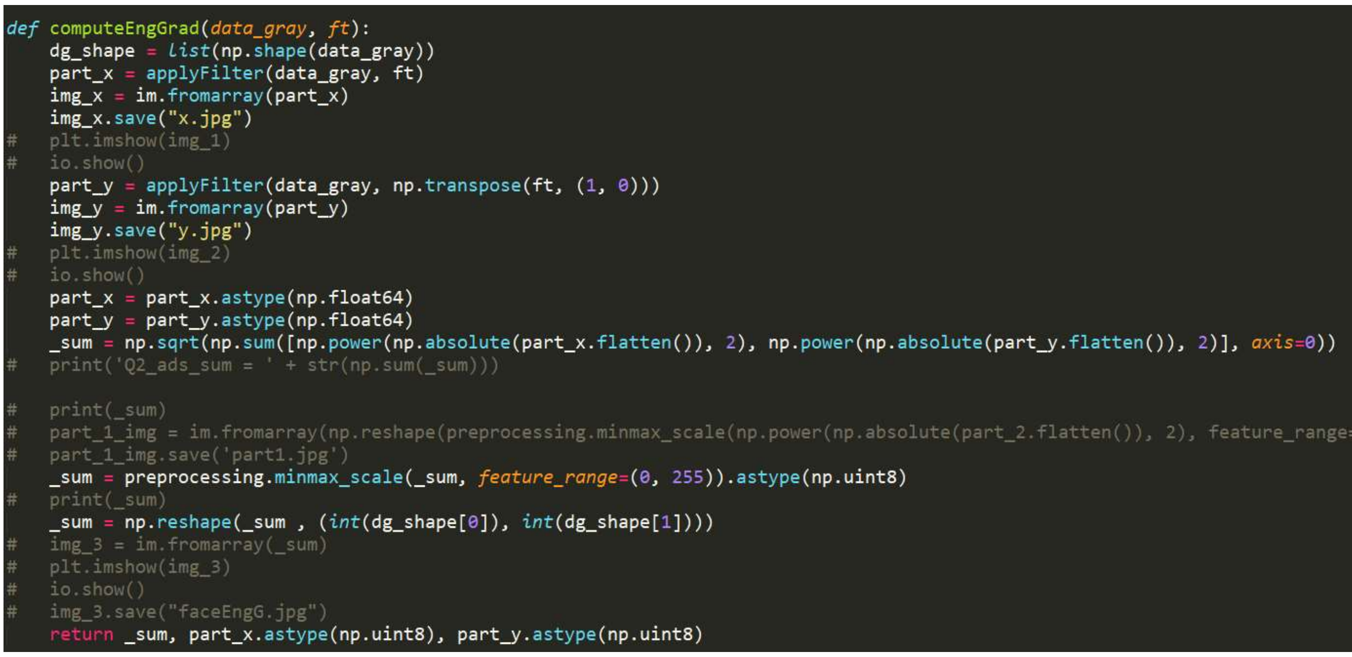
將各個 channel flatten 方便加總

計算三個 channel 個 pixel 值得平均為灰階圖的值

Reshape data 回原本的高寬

Return 灰階圖

computeEngGrad(data_gray, ft):



```
def computeEngGrad(data_gray, ft):
    dg_shape = list(np.shape(data_gray))
    part_x = applyFilter(data_gray, ft)
    img_x = im.fromarray(part_x)
    img_x.save("x.jpg")
#   plt.imshow(img_1)
#   io.show()
    part_y = applyFilter(data_gray, np.transpose(ft, (1, 0)))
    img_y = im.fromarray(part_y)
    img_y.save("y.jpg")
#   plt.imshow(img_2)
#   io.show()
    part_x = part_x.astype(np.float64)
    part_y = part_y.astype(np.float64)
    _sum = np.sqrt(np.sum([np.power(np.absolute(part_x.flatten()), 2), np.power(np.absolute(part_y.flatten()), 2)], axis=0))
#   print('Q2_ads_sum = ' + str(np.sum(_sum)))

#   print(_sum)
#   part_1_img = im.fromarray(np.reshape(preprocessing.minmax_scale(np.power(np.absolute(part_2.flatten()), 2), feature_range=
#   part_1_img.save('part1.jpg')
    _sum = preprocessing.minmax_scale(_sum, feature_range=(0, 255)).astype(np.uint8)
#   print(_sum)
    _sum = np.reshape(_sum , (int(dg_shape[0]), int(dg_shape[1])))
#   img_3 = im.fromarray(_sum)
#   plt.imshow(img_3)
#   io.show()
#   img_3.save("faceEngG.jpg")
    return _sum, part_x.astype(np.uint8), part_y.astype(np.uint8)
```

功能 ： 計算灰階圖的梯度

傳入一張灰階圖，和 filter

先算出 x, y 的灰階梯度圖(同一個 filter 運用轉置)

將算出來灰階梯度圖格式轉成 float64 和 flatten 方便運算

將 x, y 灰階梯度圖套入下式計算出整張圖灰階梯度圖

$$eng = sqrt(|F \otimes imG|^2 + |F^T \otimes imG|^2)$$

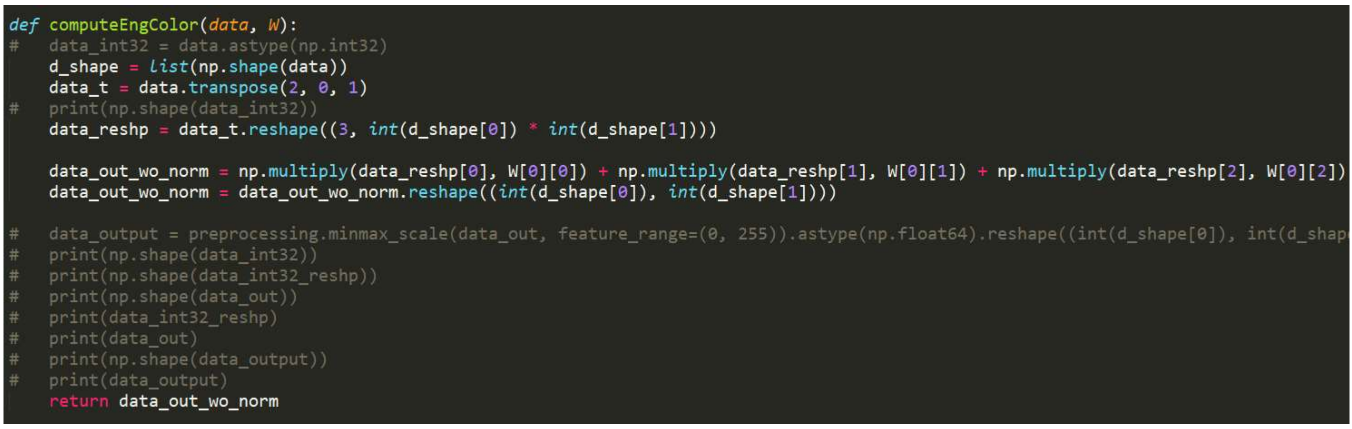
用 sklearn lib. 將資料數值正規化至 0-255

將資料轉回原本高寬

Return 灰階圖的梯度圖

Problem3

computeEngColor(data, W):



```python
def computeEngColor(data, W):
#   data_int32 = data.astype(np.int32)
    d_shape = list(np.shape(data))
    data_t = data.transpose(2, 0, 1)
#   print(np.shape(data_int32))
    data_reshp = data_t.reshape((3, int(d_shape[0]) * int(d_shape[1])))

    data_out_wo_norm = np.multiply(data_reshp[0], W[0][0]) + np.multiply(data_reshp[1], W[0][1]) + np.multiply(data_reshp[2], W[0][2])
    data_out_wo_norm = data_out_wo_norm.reshape((int(d_shape[0]), int(d_shape[1])))

#   data_output = preprocessing.minmax_scale(data_out, feature_range=(0, 255)).astype(np.float64).reshape((int(d_shape[0]), int(d_shap
#   print(np.shape(data_int32))
#   print(np.shape(data_int32_reshp))
#   print(np.shape(data_out))
#   print(data_int32_reshp)
#   print(data_out)
#   print(np.shape(data_output))
#   print(data_output)
    return data_out_wo_norm
```

功能 ： 計算彩圖能量

傳入一張 color image, 3-channel 對應 weight

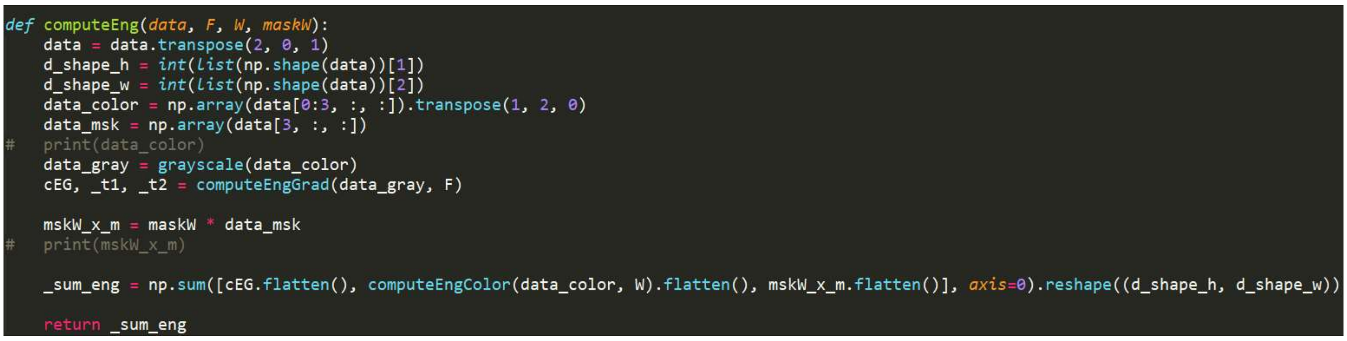
將圖片 shape (h, w, c) transpose to (c, h, w)方便使用

實作下式求得彩圖能量圖

$$eng = computeEngGrad(im4(:,:,1:3), F) + computeEngColor(im4(:,:,1:3), W) + maskW * im4(:,:,4)$$

Return 彩圖能量圖

computeEng(data, F, W, maskW):



```python
def computeEng(data, F, W, maskW):
    data = data.transpose(2, 0, 1)
    d_shape_h = int(list(np.shape(data))[1])
    d_shape_w = int(list(np.shape(data))[2])
    data_color = np.array(data[0:3, :, :]).transpose(1, 2, 0)
    data_msk = np.array(data[3, :, :])
#   print(data_color)
    data_gray = grayscale(data_color)
    cEG, _t1, _t2 = computeEngGrad(data_gray, F)

    mskW_x_m = maskW * data_msk
#   print(mskW_x_m)

    _sum_eng = np.sum([cEG.flatten(), computeEngColor(data_color, W).flatten(), mskW_x_m.flatten()], axis=0).reshape((d_shape_h, d_shape_w))

    return _sum_eng
```

功能 ： 計算能量圖

傳入一張彩色圖

將圖片 shape (h, w, c) transpose to (c, h, w)方便使用

各 RGB channel 和 mask 部分拆開

RGB 圖轉置回(hwc)

調用 grayscale 返回灰階圖

調用 computEngGrad 返回梯度能量圖

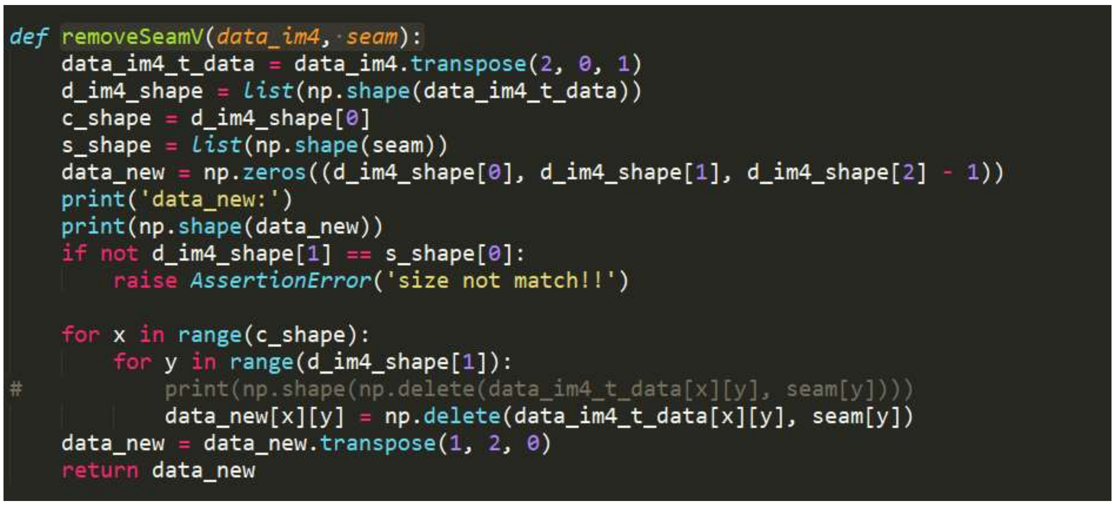
Mask * MaskWeight

Flatten 方便運算

計算出能量圖

Return data

Problem4
removeSeamV(data_im4, seam):



```python
def removeSeamV(data_im4, seam):
    data_im4_t_data = data_im4.transpose(2, 0, 1)
    d_im4_shape = list(np.shape(data_im4_t_data))
    c_shape = d_im4_shape[0]
    s_shape = list(np.shape(seam))
    data_new = np.zeros((d_im4_shape[0], d_im4_shape[1], d_im4_shape[2] - 1))
    print('data_new:')
    print(np.shape(data_new))
    if not d_im4_shape[1] == s_shape[0]:
        raise AssertionError('size not match!!')

    for x in range(c_shape):
        for y in range(d_im4_shape[1]):
#               print(np.shape(np.delete(data_im4_t_data[x][y], seam[y])))
            data_new[x][y] = np.delete(data_im4_t_data[x][y], seam[y])
    data_new = data_new.transpose(1, 2, 0)
    return data_new
```

功能 ： 移除梯度低的位置的 pixel 來縮小長寬

傳入圖片(RGBM)和 seam(1-d)

先將圖片 shape (h, w, c) transpose to (c, h, w)方便使用

獲取圖片 shape

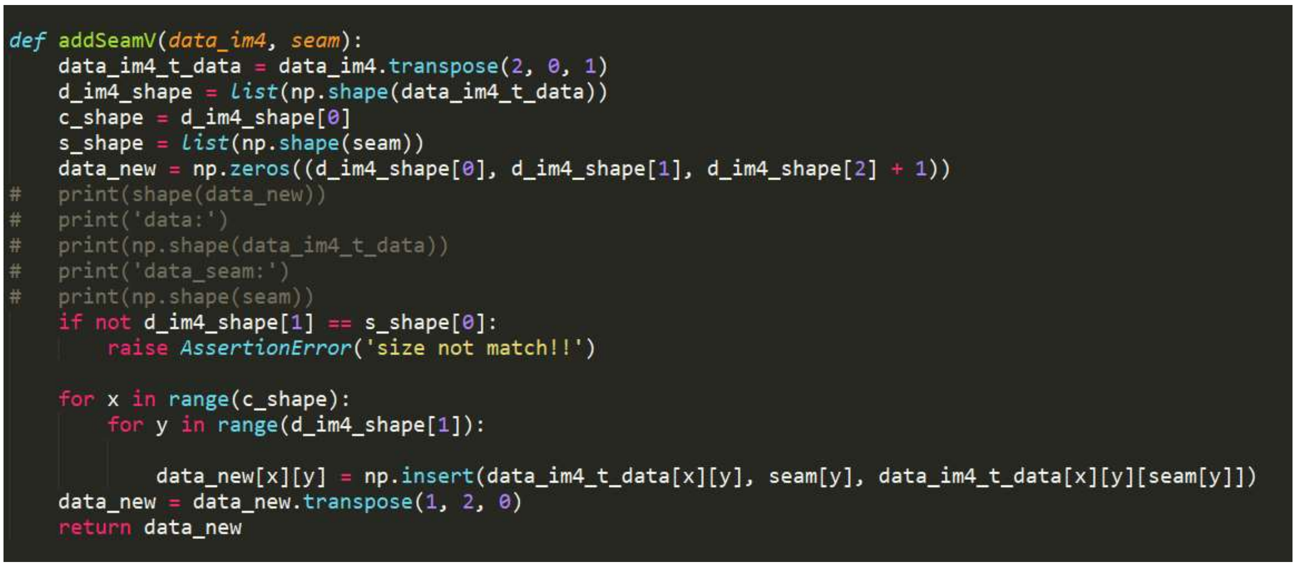
判圖片高是否等同 seam 的長度，不相等停止運行，並且提醒錯誤

循序將各個 channel 圖片 seam 處減少 pixel

完成所有 channel 後將得到的 data 轉置回圖片標準(h, w, c)

傳回 data

addSeamV(data_im4, seam):



```python
def addSeamV(data_im4, seam):
    data_im4_t_data = data_im4.transpose(2, 0, 1)
    d_im4_shape = list(np.shape(data_im4_t_data))
    c_shape = d_im4_shape[0]
    s_shape = list(np.shape(seam))
    data_new = np.zeros((d_im4_shape[0], d_im4_shape[1], d_im4_shape[2] + 1))
#   print(shape(data_new))
#   print('data:')
#   print(np.shape(data_im4_t_data))
#   print('data_seam:')
#   print(np.shape(seam))
    if not d_im4_shape[1] == s_shape[0]:
        raise AssertionError('size not match!!')

    for x in range(c_shape):
        for y in range(d_im4_shape[1]):

            data_new[x][y] = np.insert(data_im4_t_data[x][y], seam[y], data_im4_t_data[x][y][seam[y]])
    data_new = data_new.transpose(1, 2, 0)
    return data_new
```

功能 ： 在梯度低的位置加入 pixel 來放大長寬

傳入圖片(RGBM)和 seam(1-d)

先將圖片 shape (h, w, c) transpose to (c, h, w)方便使用

獲取圖片 shape

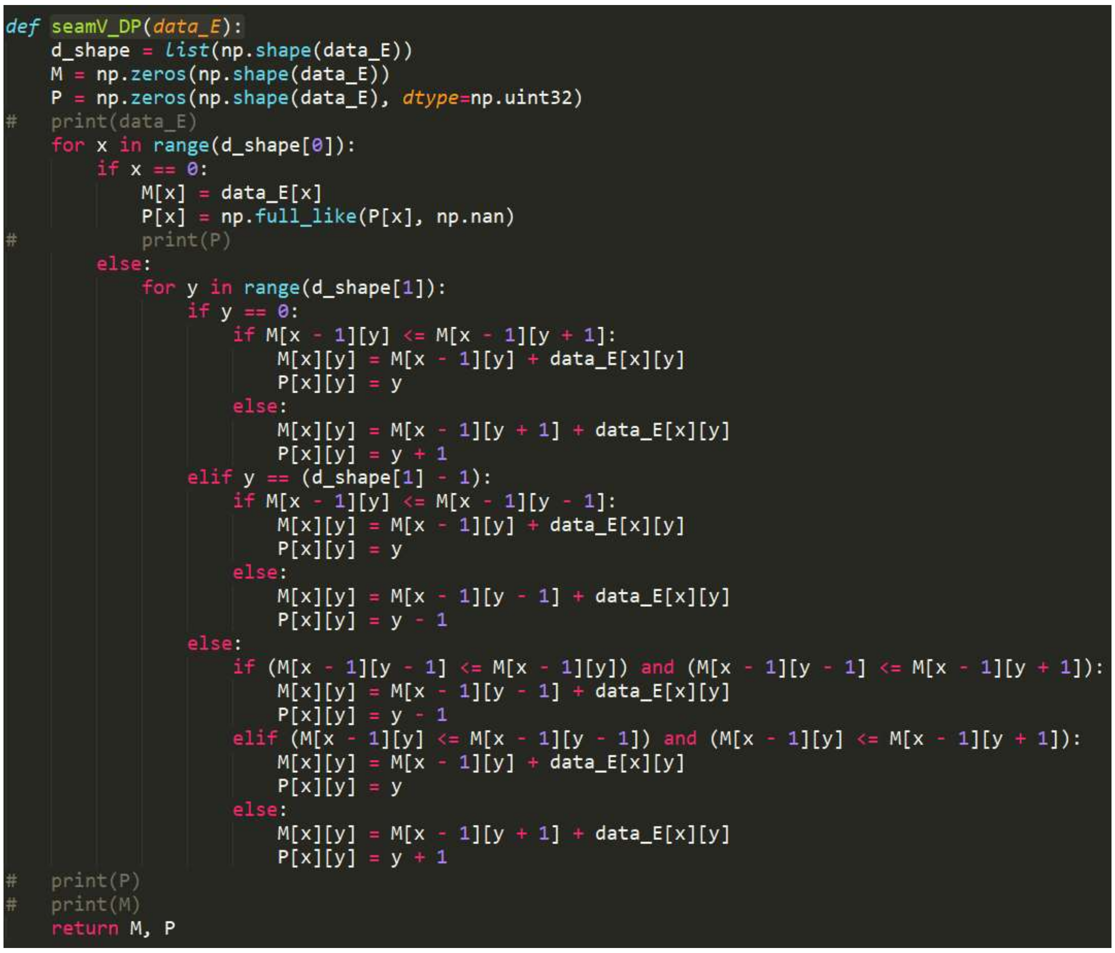
判圖片高是否等同 seam 的長度，不相等停止運行，並且提醒錯誤

循序將各個 channel 圖片 seam 處增加 pixel(插入附近 pixel 平均值)

完成所有 channel 後將得到的 data 轉置回圖片標準(h, w, c)

傳回 data

seamV_DP(data_E):



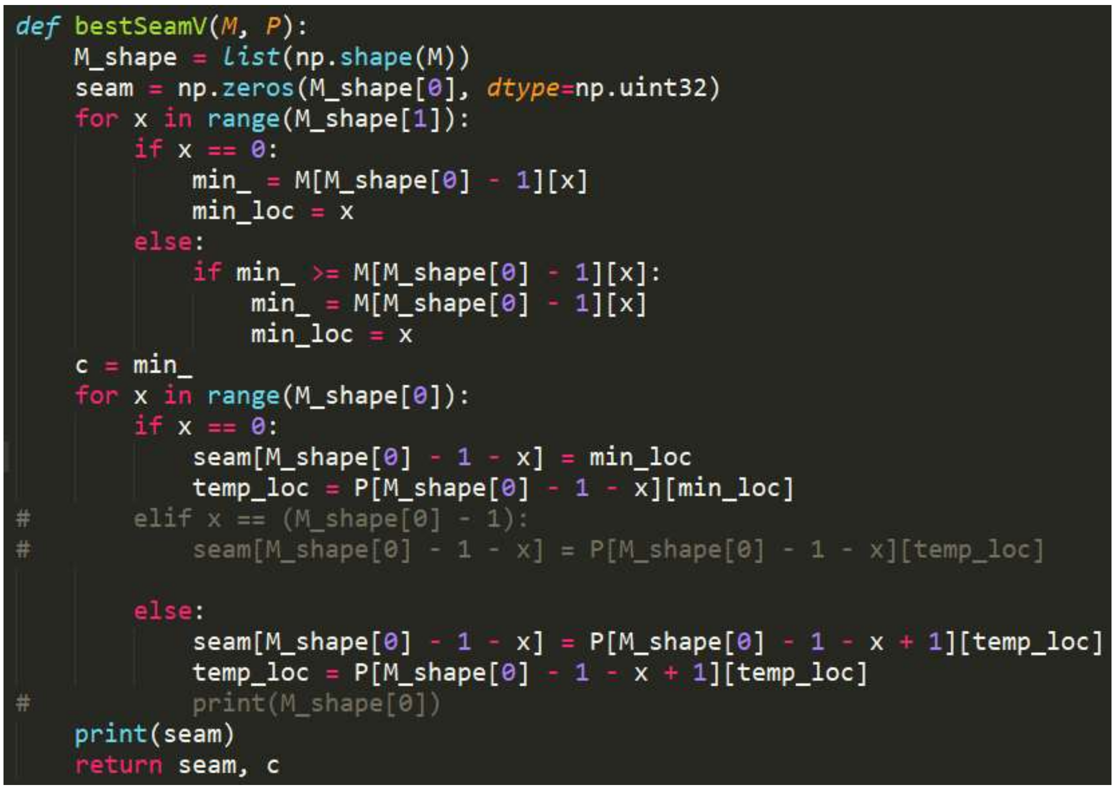
```
def seamV_DP(data_E):
    d_shape = list(np.shape(data_E))
    M = np.zeros(np.shape(data_E))
    P = np.zeros(np.shape(data_E), dtype=np.uint32)
#   print(data_E)
    for x in range(d_shape[0]):
        if x == 0:
            M[x] = data_E[x]
            P[x] = np.full_like(P[x], np.nan)
#           print(P)
        else:
            for y in range(d_shape[1]):
                if y == 0:
                    if M[x - 1][y] <= M[x - 1][y + 1]:
                        M[x][y] = M[x - 1][y] + data_E[x][y]
                        P[x][y] = y
                    else:
                        M[x][y] = M[x - 1][y + 1] + data_E[x][y]
                        P[x][y] = y + 1
                elif y == (d_shape[1] - 1):
                    if M[x - 1][y] <= M[x - 1][y - 1]:
                        M[x][y] = M[x - 1][y] + data_E[x][y]
                        P[x][y] = y
                    else:
                        M[x][y] = M[x - 1][y - 1] + data_E[x][y]
                        P[x][y] = y - 1
                else:
                    if (M[x - 1][y - 1] <= M[x - 1][y]) and (M[x - 1][y - 1] <= M[x - 1][y + 1]):
                        M[x][y] = M[x - 1][y - 1] + data_E[x][y]
                        P[x][y] = y - 1
                    elif (M[x - 1][y] <= M[x - 1][y - 1]) and (M[x - 1][y] <= M[x - 1][y + 1]):
                        M[x][y] = M[x - 1][y] + data_E[x][y]
                        P[x][y] = y
                    else:
                        M[x][y] = M[x - 1][y + 1] + data_E[x][y]
                        P[x][y] = y + 1
#   print(P)
#   print(M)
    return M, P
```

功能 ： 利用 dynamic programing 方法來算出累積 cost 矩陣 M 和路徑來源矩陣 P

傳入能量圖

Return 累積 cost 矩陣 M 和路徑來源矩陣 P

bestSeamV(M, P):



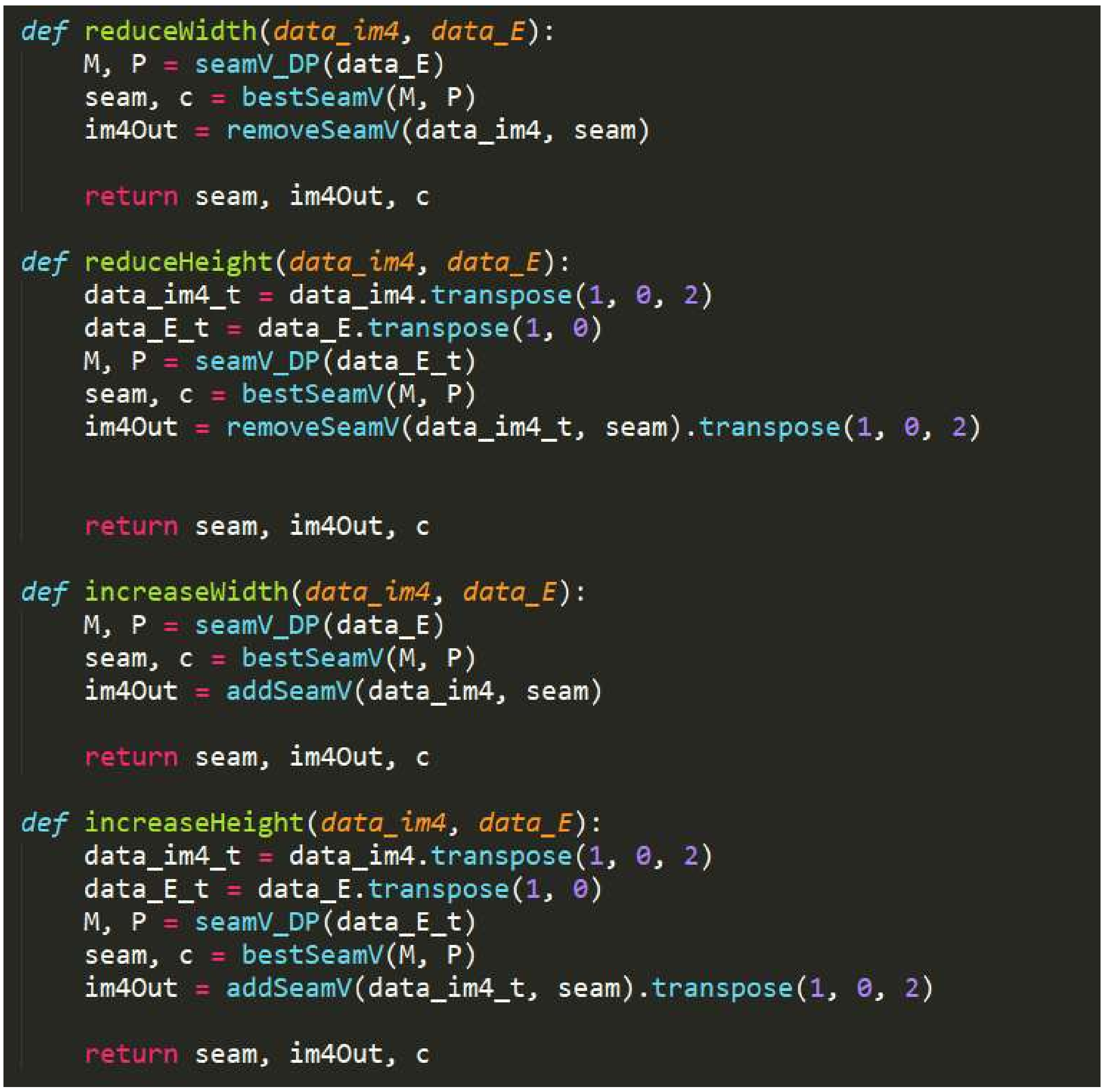
```
def bestSeamV(M, P):
    M_shape = list(np.shape(M))
    seam = np.zeros(M_shape[0], dtype=np.uint32)
    for x in range(M_shape[1]):
        if x == 0:
            min_ = M[M_shape[0] - 1][x]
            min_loc = x
        else:
            if min_ >= M[M_shape[0] - 1][x]:
                min_ = M[M_shape[0] - 1][x]
                min_loc = x
    c = min_
    for x in range(M_shape[0]):
        if x == 0:
            seam[M_shape[0] - 1 - x] = min_loc
            temp_loc = P[M_shape[0] - 1 - x][min_loc]
#       elif x == (M_shape[0] - 1):
#           seam[M_shape[0] - 1 - x] = P[M_shape[0] - 1 - x][temp_loc]

        else:
            seam[M_shape[0] - 1 - x] = P[M_shape[0] - 1 - x + 1][temp_loc]
            temp_loc = P[M_shape[0] - 1 - x + 1][temp_loc]
#           print(M_shape[0])
    print(seam)
    return seam, c
```

功能 ：利用累積 cost 矩陣 M 和路徑來源矩陣 P 算出最好的 seam

傳入積 cost 矩陣 M 和路徑來源矩陣 P

Return seam

reduceWidth(data_im4, data_E):
reduceHeight(data_im4, data_E):
increaseWidth(data_im4, data_E):
increaseHeight(data_im4, data_E):

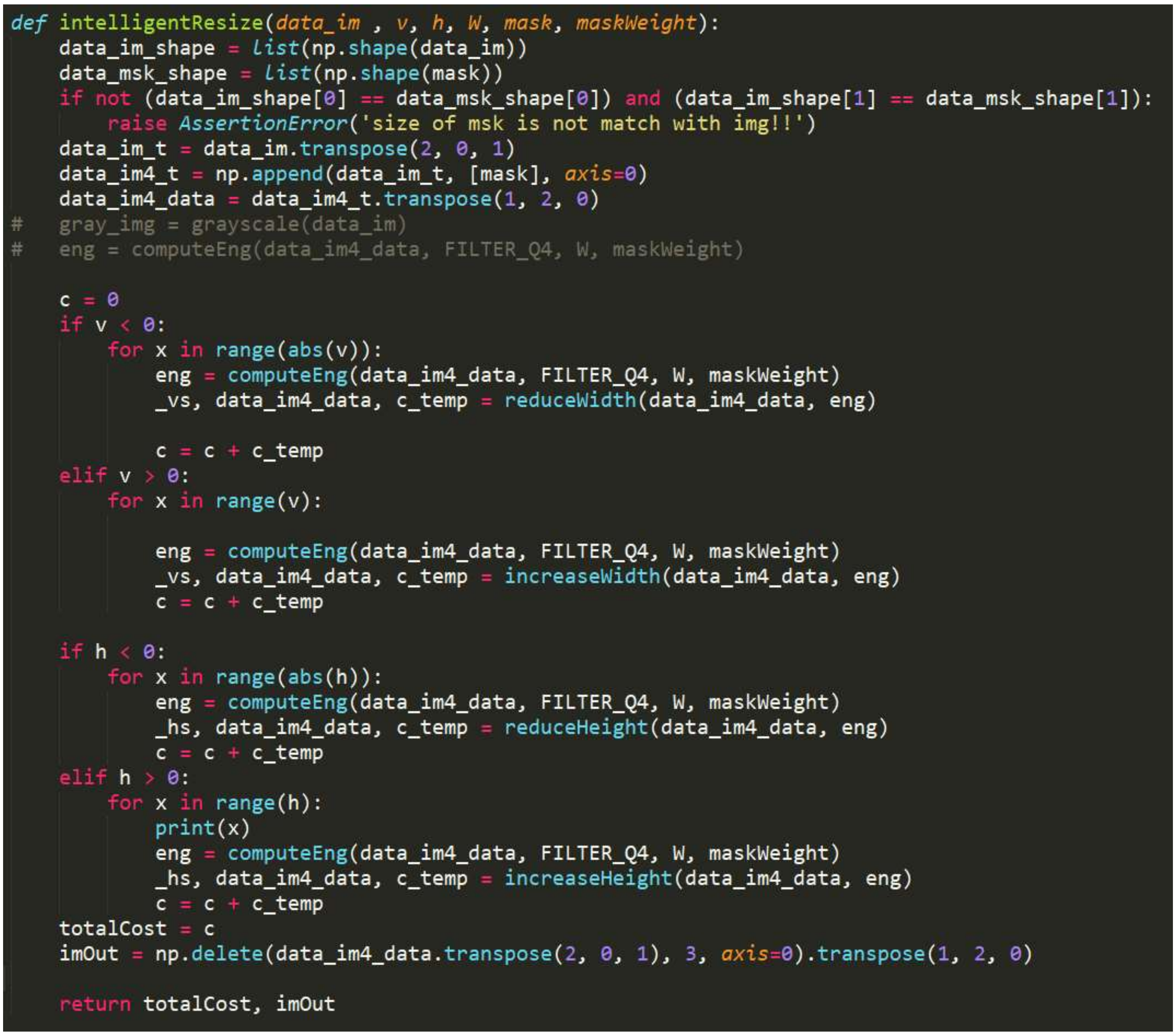


```python
def reduceWidth(data_im4, data_E):
    M, P = seamV_DP(data_E)
    seam, c = bestSeamV(M, P)
    im4Out = removeSeamV(data_im4, seam)

    return seam, im4Out, c

def reduceHeight(data_im4, data_E):
    data_im4_t = data_im4.transpose(1, 0, 2)
    data_E_t = data_E.transpose(1, 0)
    M, P = seamV_DP(data_E_t)
    seam, c = bestSeamV(M, P)
    im4Out = removeSeamV(data_im4_t, seam).transpose(1, 0, 2)


    return seam, im4Out, c

def increaseWidth(data_im4, data_E):
    M, P = seamV_DP(data_E)
    seam, c = bestSeamV(M, P)
    im4Out = addSeamV(data_im4, seam)

    return seam, im4Out, c

def increaseHeight(data_im4, data_E):
    data_im4_t = data_im4.transpose(1, 0, 2)
    data_E_t = data_E.transpose(1, 0)
    M, P = seamV_DP(data_E_t)
    seam, c = bestSeamV(M, P)
    im4Out = addSeamV(data_im4_t, seam).transpose(1, 0, 2)

    return seam, im4Out, c
```

功能 ： 增加減少寬高的 pixel 數量

Return seam, 輸出圖片 , totalCost

intelligentResize(data_im , v, h, W, mask, maskWeight):



```python
def intelligentResize(data_im , v, h, W, mask, maskWeight):
    data_im_shape = list(np.shape(data_im))
    data_msk_shape = list(np.shape(mask))
    if not (data_im_shape[0] == data_msk_shape[0]) and (data_im_shape[1] == data_msk_shape[1]):
        raise AssertionError('size of msk is not match with img!!')
    data_im_t = data_im.transpose(2, 0, 1)
    data_im4_t = np.append(data_im_t, [mask], axis=0)
    data_im4_data = data_im4_t.transpose(1, 2, 0)
#   gray_img = grayscale(data_im)
#   eng = computeEng(data_im4_data, FILTER_Q4, W, maskWeight)

    c = 0
    if v < 0:
        for x in range(abs(v)):
            eng = computeEng(data_im4_data, FILTER_Q4, W, maskWeight)
            _vs, data_im4_data, c_temp = reduceWidth(data_im4_data, eng)

            c = c + c_temp
    elif v > 0:
        for x in range(v):

            eng = computeEng(data_im4_data, FILTER_Q4, W, maskWeight)
            _vs, data_im4_data, c_temp = increaseWidth(data_im4_data, eng)
            c = c + c_temp

    if h < 0:
        for x in range(abs(h)):
            eng = computeEng(data_im4_data, FILTER_Q4, W, maskWeight)
            _hs, data_im4_data, c_temp = reduceHeight(data_im4_data, eng)
            c = c + c_temp
    elif h > 0:
        for x in range(h):
            print(x)
            eng = computeEng(data_im4_data, FILTER_Q4, W, maskWeight)
            _hs, data_im4_data, c_temp = increaseHeight(data_im4_data, eng)
            c = c + c_temp
    totalCost = c
    imOut = np.delete(data_im4_data.transpose(2, 0, 1), 3, axis=0).transpose(1, 2, 0)

    return totalCost, imOut
```

功能 : Seam Carving 演算法

傳入 image, 縮減寬量, 縮減高量, mask, maskWeight

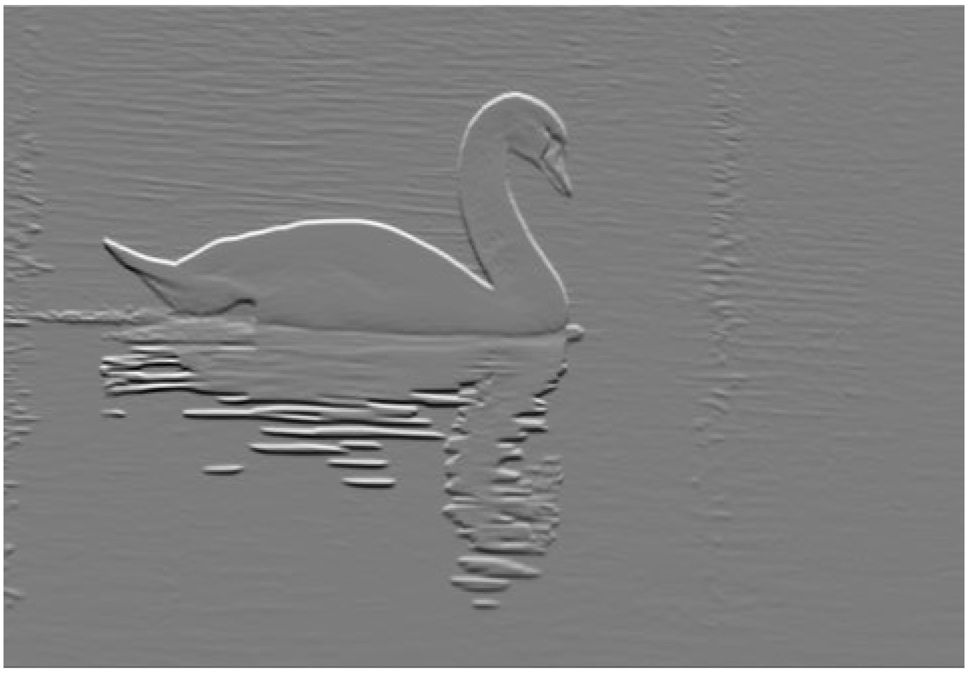
判斷寬高是增加縮減並做對應的處理

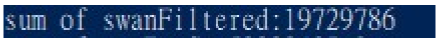
Return seam carving 後的圖

Problem5
Time is not enough!!!

三、 執行結果

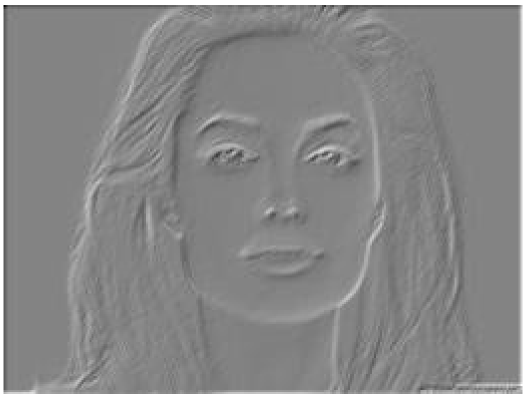
Problem 1：

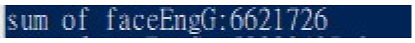




```
sum of swanFiltered:19729786
```

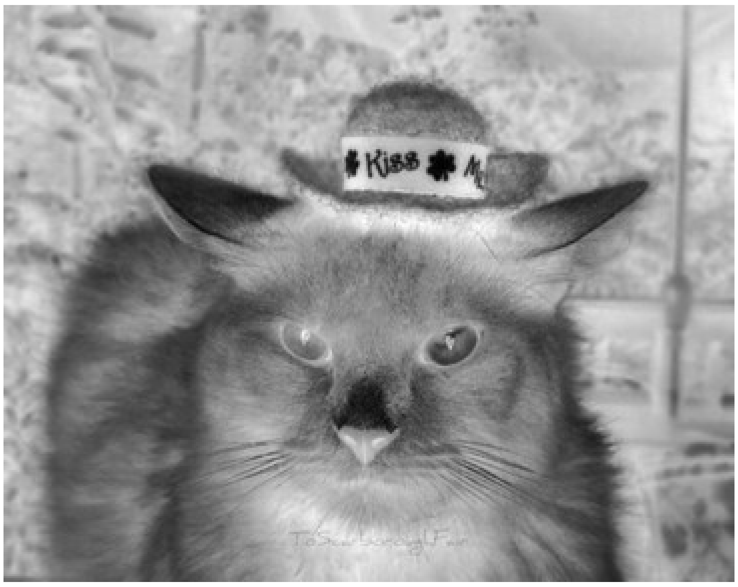
Problem 2：

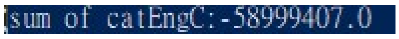




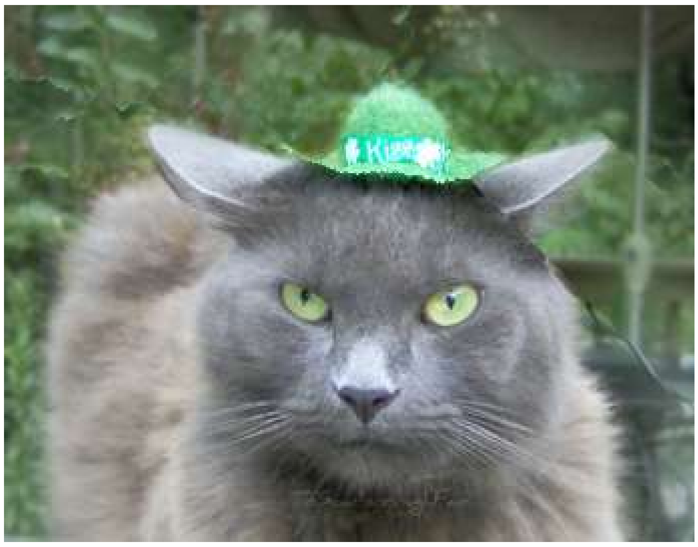
```
sum of faceEngG:6621726
```

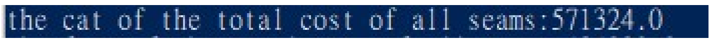
Problem 3：

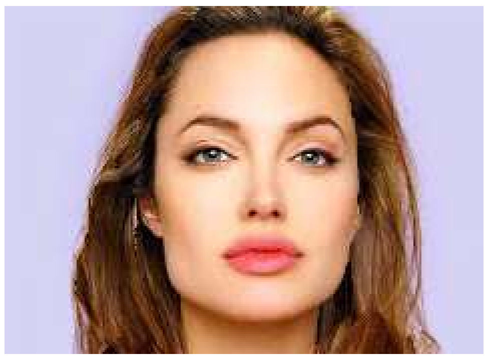


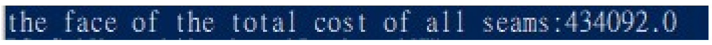

```
sum of catEngC:-58999407.0
```

Problem 4：




```
the cat of the total cost of all seams:571324.0
```




```
the face of the total cost of all seams:434092.0
```

Problem 5 :time is not engough