

國立清華大學
計算機視覺
Computer Vision



國立清華大學
NATIONAL TSING HUA UNIVERSITY

Homework 1

系所級:電子所二年級

學號:111063548

姓名:蕭方凱

指導老師:孫民教授

目錄

1.	Implementation	3
1.1	Image filtering.....	3
1.2	Extract and combine the high-frequency and low-frequency signals.....	5
1.3	Others.....	6
2.	Experiments	6
2.1	Hybrid image	6
2.2	Other hybrid images.....	7
2.3	Customized hybrid images.....	9
3.	Discussion	10

1. Implementation

1.1 Image filtering

```
42 # ===== Start OF YOUR CODE =====
43 output = np.zeros_like(image)
44
45 # print("image shape:", image.shape)
46 # print("imfilter shape:", imfilter.shape)
47 # print(image)
48
49 kernel_height, kernel_width = imfilter.shape
50 image_height, image_width, depth = image.shape
51
52 # 不擴充陣列下卷積後之輸出size
53 output_height = image_height - kernel_height + 1
54 output_width = image_width - kernel_width + 1
55 # print(output_height, output_width)
56 # print(pad_adding_height, pad_adding_width)
57
58 # 擴充pad大小
59 pad_adding_height = int((kernel_height - 1) / 2)
60 pad_adding_width = int((kernel_width - 1) / 2)
61
62 # padding image
63 pad_image = np.pad(image,
64                     ((pad_adding_height, pad_adding_width), (pad_adding_height, pad_adding_width), (0, 0)),
65                     "constant",
66                     constant_values=(0, 0))
67 # print(pad_image.shape)
68
69 for i in range(image_height):
70     for j in range(image_width):
71         for d in range(depth):
72             input_slice = pad_image[i:i+kernel_height, j:j+kernel_width, d]
73             output[i, j, d] = np.sum(input_slice * imfilter)
74
75
76 # ===== END OF YOUR CODE =====
```

Fig 1 my_imfilter.py

首先，先透過 `imfilter.shape` 及 `image.shape` 得到兩陣列的大小，前者為卷積核 `kernel`，後者為輸入照片。假設有一 3×3 (height x width) 的卷積核要對 5×5 , 7×7 , 9×9 (height x width) 的圖像分別進行卷積運算，如下圖：

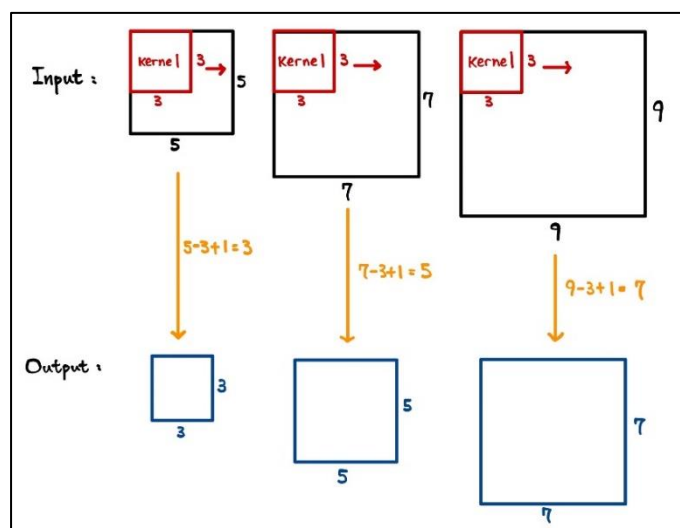


Fig 2 卷積運算輸入與輸出

一個 $N \times N$ 的 image 與 $n \times n$ 的 kernel 進行卷積運算後，輸出 size 為 $N-n+1$

可以觀察出對三張照片進行卷積運算後之輸出大小分別為 3x3, 5x5, 7x7，皆與原輸入照片大小不同，故須進行 padding，使用 np.pad 去將 image 矩陣擴增，擴增大小與 kernel 有關，可以從上方例子觀察出需在 height 及 width 處分別需擴增元素數量的公式為：

$$\text{image height 需擴增數量} : \frac{\text{kernel height} - 1}{2}$$

$$\text{image width 需擴增數量} : \frac{\text{kernel width} - 1}{2}$$

擴增完以後，再將新的 pad_image 放入 for 迴圈，將其切片與 kernel 相乘，去做 sliding 運算的功能。

Sliding 運算示意圖：

```

69  for i in range(image_height):
70      for j in range(image_width):
71          for d in range(depth):
72              input_slice = pad_image[i:i+kernel_height, j:j+kernel_width, d]
73              output[i, j, d] = np.sum(input_slice * imfilter)

```

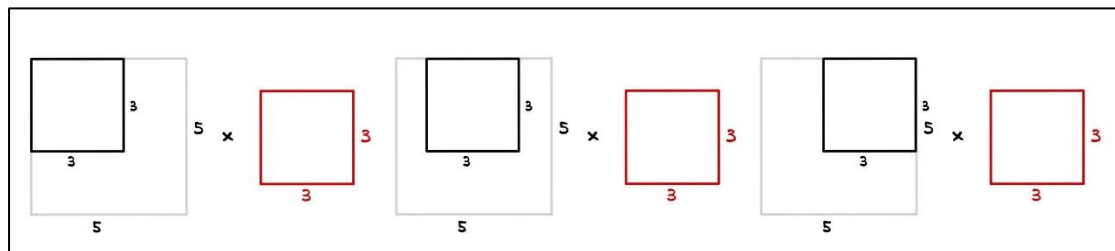


Fig 3 for 迴圈之($i = 0, j = 0 \sim 2$)

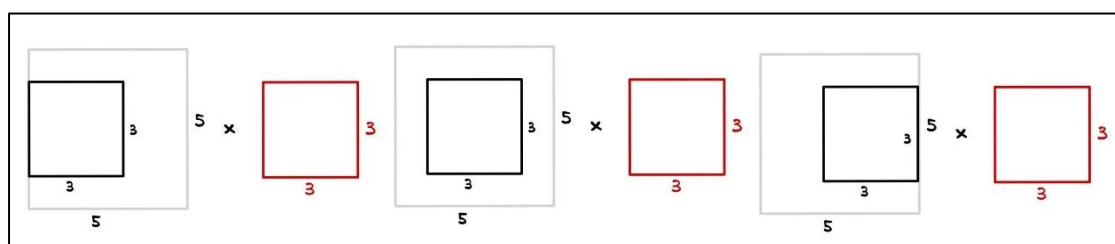


Fig 4 for 迴圈之($i = 1, j = 0 \sim 2$)

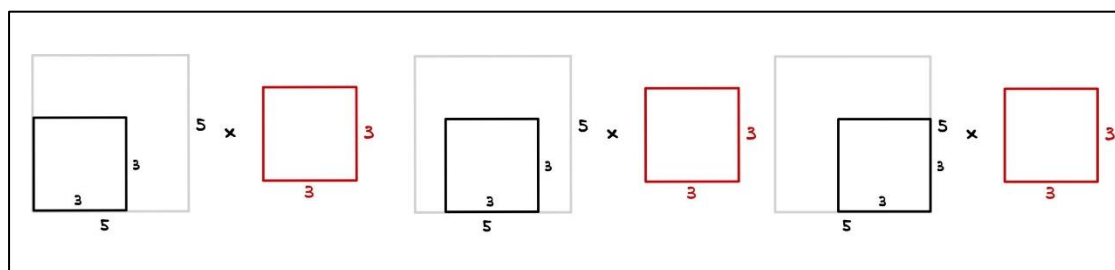


Fig 5 for 迴圈之($i = 2, j = 0 \sim 2$)

1.2 Extract and combine the high-frequency and low-frequency signals

Gaussian blur coding (teacher's example):

```

29 # =====
30 # === Filtering and Hybrid Image construction ===
31 # =====
32 cutoff_frequency = 7 # This is the standard deviation, in pixels, of the
33 # Gaussian blur that will remove the high frequencies from one image and
34 # remove the low frequencies from another image (by subtracting a blurred
35 # version from the original version). You will want to tune this for every
36 # image pair to get the best results.
37 gaussian_filter = gauss2D(shape=(cutoff_frequency*4+1,cutoff_frequency*4+1), sigma = cutoff_frequency)

```

Blurring image1 and image2 to get the low frequencies image:

```

48 low_frequencies = my_imfilter(image1, gaussian_filter) # You need to modify here
49 low_frequencies_cat = my_imfilter(image2, gaussian_filter)

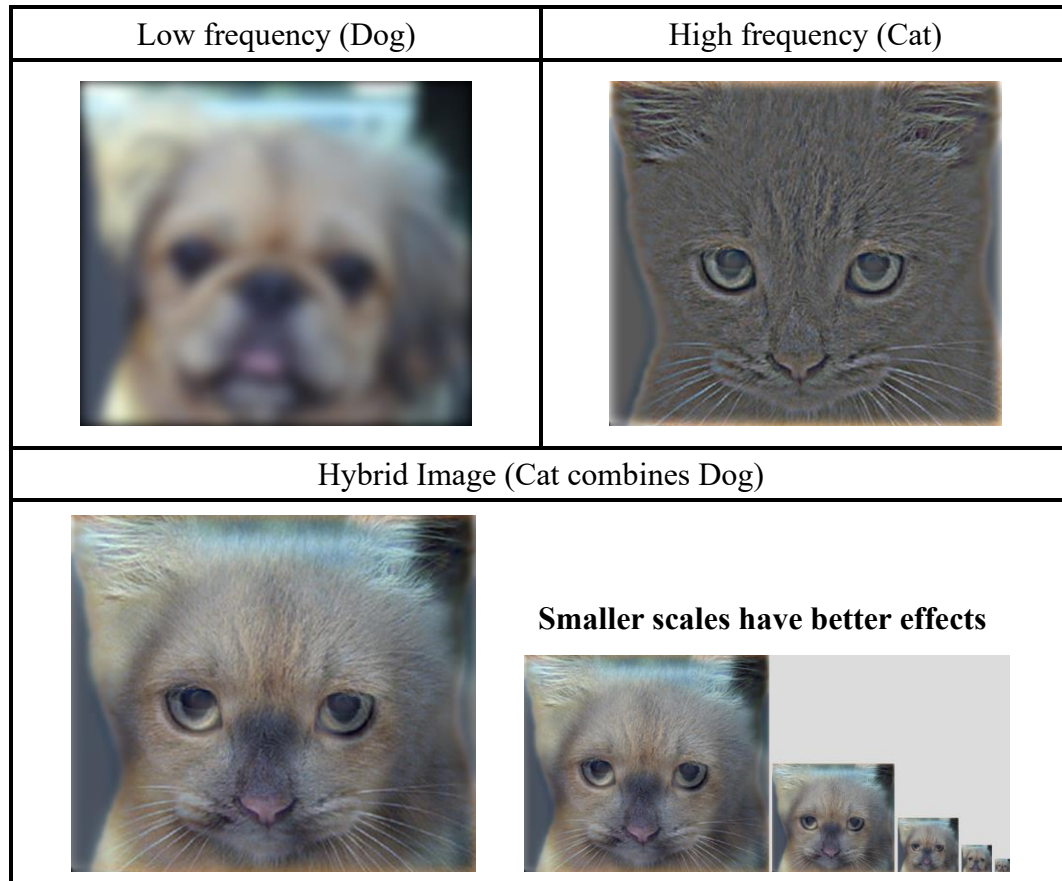
```

Get image2's high frequency result by subtracting low frequencies_cat from image2:

```

56 high_frequencies = image2 - low_frequencies_cat # You need to modify here
57 # laplacian_filter = np.array([[0, 1, 0],
58 #                               [1, -4, 1],
59 #                               [0, 1, 0]])
60 # high_frequencies = my_imfilter(image2, laplacian_filter)

```

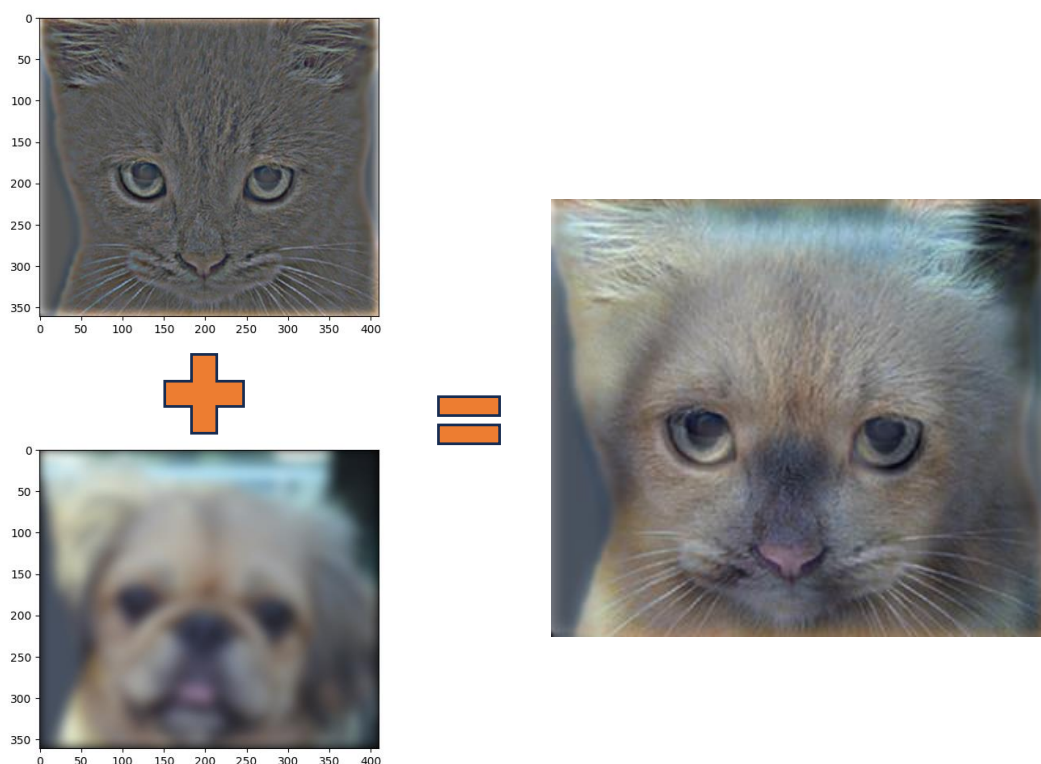


1.3 Others

There's no other package required in my implementation; I only used the packages from teacher's program examples.

2. Experiments

2.1 Hybrid image



利用 hwl.py 之 gaussian_filter

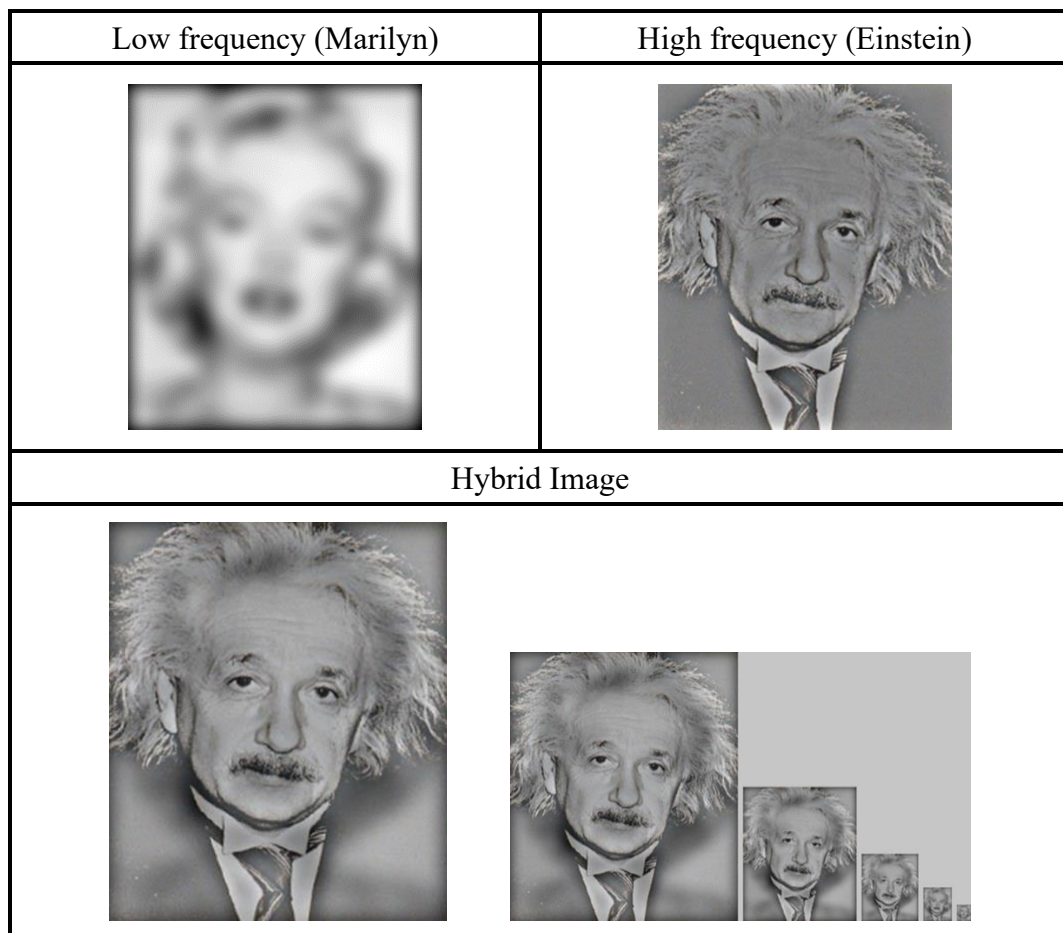
```
gaussian_filter =  
gauss2D(shape=(cutoff_frequency*4+1,cutoff_frequency*4+1),  
sigma = cutoff_frequency)
```

對貓跟狗的原圖進行模糊處理，得到貓狗的 low_frequency 結果，接著取狗的 low_frequency，而貓咪則是利用 low_frequency 再加以計算，將貓咪原圖減去貓咪的 low_frequency，得到貓咪的 high_frequency，如上圖。

最後將 low_frequency(狗)+high_frequency(貓)，得到合併後的圖，如上方右圖。

2.2 Other hybrid images

Trying to combine Einstein and Marilyn, the results are shown as follows:



合成效果不如 2.1 的貓狗，在 scale 較大時看不出 Marilyn 的特徵，需調整 gaussian filter 的內部參數。

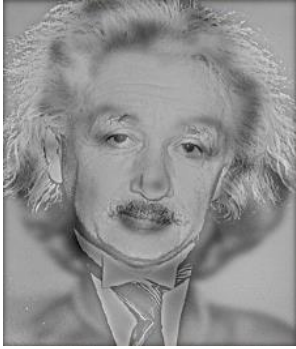
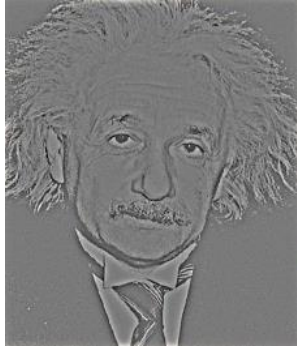
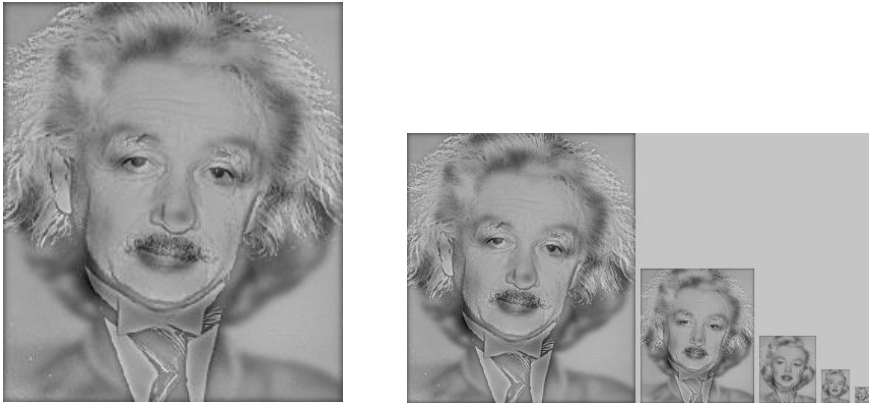
因上圖 einstein 的特徵過於強烈，故須減少高頻部分的信息，透過縮小 gaussian filter 的尺寸去實現，以濾除過多高頻信息，將 einstein 的特徵弱化。

```
32 cutoff_frequency = 4.5 # This is the standard deviation, in pixels, of the
33 # Gaussian blur that will remove the high frequencies from one image and
34 # remove the low frequencies from another image (by subtracting a blurred
35 # version from the original version). You will want to tune this for every
36 # image pair to get the best results.
37 gaussian_filter = gauss2D(shape=(cutoff_frequency*4+1,cutoff_frequency*4+1), sigma = cutoff_frequency-2)
```

將 cutoff_frequency 調小使 filter 尺寸變小，本題調整成 4.5，kernel size 變成 19 x 19，另外也對 sigma 做些微變化，使照片合成效果提升。

(優化後照片如下頁)

Trying to combine better, the results are shown as follows:



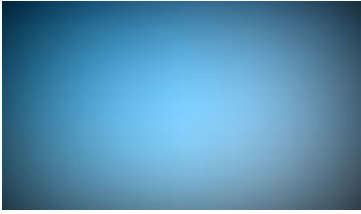
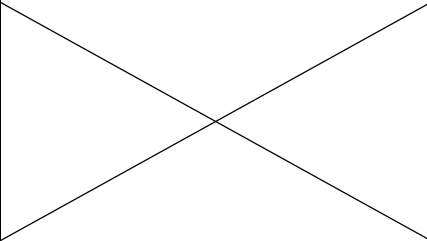
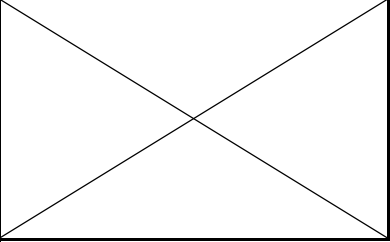


Low frequency (Marilyn)	High frequency (Einstein)
	
Hybrid Image	
	

比起上頁成果，本頁的合成結果 Marilyn 的特徵更加強烈，使照片不需要 scale 到很小，也可擁有上頁照片 scale 到很小的效果。

2.3 Customized hybrid images

本題將夕陽下的山景與白天的天空合併，合併結果為早晨日出的山景。

透過修改 cutoff frequency 參數，避免高頻部分特徵過於強烈導致合成照片偏向某一張照片。

	Sky	Mountain
Original Images		
Low frequency Images		
High frequency Images		
Hybrid Images		

3. Discussion

每組 pair pictures 都有自己最適合的 gaussian filter size，老師給的範例程式之 gaussian filter 針對合併貓狗 pair 表現非常好，而其他組合皆須微調參數才可使照片不過於貼近 high frequency 的特徵，主要調整方法為縮小 filter size，即可弱化高頻影像的特徵，同時增強低頻影像的特徵，使合成結果更貼近兩者，得到不錯的平衡。

對照片影像做低頻處理，可應用在一些交友軟體，不讓使用者先看到清楚的臉，防止外貌資訊量太多忽略了聊天的重要性，也可應用在一些經模糊處理的特效；。而套用 sobel filter 到照片上，有分 vertical 及 horizontal 之矩陣形式，vertical sobel 可使照片呈現浮雕感，針對一些照片(如貓咪)，vertical sobel 也更能看清照片的輪廓。

合成照片的部分，兩照片的尺寸需一模一樣才可進行合成，找到最佳平衡的過程需不斷調整 filter size，另外針對取得高頻影像的部分，有兩種方法，第一種為先透過 gaussian filter 得到該照片的低頻部分，再將原照片減去其低頻部分，即可得到高頻部分，第二種方法為使用 laplacian filter，使用老師提供的矩陣如下：

$$laplacian\ filter = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

經實際測試後，第一種方法效果較佳，本次作業針對高頻部分皆採用先取得低頻部分再做相減得到影像的高頻部分。