
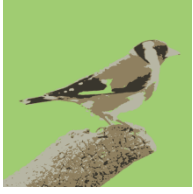


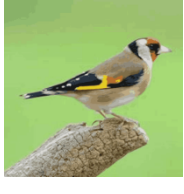

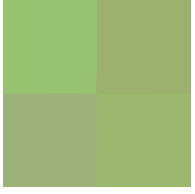





1-1.

K	2	4	8	16	32
重建圖片					

1-2.

K	2	4	8	16	32
重建圖片					

1-3.

比較 (r, g, b) 以及 (r, g, b, x, y) 的分類結果後, 可以發現有加入 x, y 座標的新資料, 在k-means中更著重使用pixel 在圖片中的位置 (x, y) 當作分類的依據, 因此使用 (r, g, b, x, y) 的資料做cluster重建後的結果會傾向將”局部”有相似色彩的pixel群聚在一起, 且 (x, y) 資料的影響比 (r, g, b) 來的大, 因此在 $k=2, 4$ 的時候, 重建後看起來像是將圖片垂直/水平對半, 或是平均等分成4個小正方形, 而在 $k=8, 16, 32$ 重建後看起來會有類似image segmentation的效果, 而使用 (r, g, b) 的資料做cluster重建後的結果會傾向將”整體”有相似色彩的pixel群聚在一起, 因此重建後視覺上會與原圖比較相近






我認為是因為, r, g, b 三個channel的值域為 $[0, 255]$, 但 x, y 的值域卻是 $[0, 1023]$, 且我在實作k-means演算法時, 是利用MSE當作相似度的判斷標準, 因此在極端狀況下, $p_1 = (0, 0, 0, 0, 0)$, $p_2 = (255, 255, 255, 1023, 1023)$ 之間的相似度為 $255^2 + 255^2 + 255^2 + 1023^2 + 1023^2$ (暫時不考慮square root以方便理解), 其中 r, g, b 所佔的部分為195075, 而 x, y 所佔的部分為2093058, 比例近乎1:10, 因此在 (r, g, b, x, y) 與 (r, g, b) 的資料對比中, 可以發現 (x, y) 對於資料有著很強的影響力

我認為的解決辦法有二：





1. 將 r, g, b, x, y 每一個channel都做normalization, 使其值域都在 $[0, 1]$, 這樣去作分類的話 r, g, b, x, y 每一個channel對於相似度都有著相同的影響力。
2. 對每一個feature加上各自的weight, ex : W_r, W_g, W_b, W_x, W_y , 便可以藉由改變weight來改變各項feature的影響力。也可以發現當

$W_r = W_g = W_b = 1/(255^2)$, $W_x = W_y = 1/(1023^2)$ 的時候, 1號方法其實就是2號方法的一個特例。

以下為使用1號方法所產生的結果

K	2	4	8	16	32
重建圖片					






2-1

ith	1	2	3	4
image				

Mean face



2-2

Reconstruct components	3	50	170	240	345
image					

2-3

n	Mean square error
3	746.799
50	236.554
170	46.717
240	13.366
345	0.215

2-4

k = 1, n = 3, test scores : [0.71666, 0.6 , 0.74166], avg : 0.686
k = 3, n = 3, test scores : [0.56666, 0.65833, 0.53333], avg : 0.586
k = 5, n = 3, test scores : [0.55 , 0.54166, 0.45833], avg : 0.51666
k = 1, n = 50, test scores : [0.91666, 0.95 , 0.925], avg : 0.93055
k = 3, n = 50, test scores : [0.9 , 0.91666, 0.9], avg : 0.90555
k = 5, n = 50, test scores : [0.79166, 0.75833, 0.83333], avg : 0.79444
k = 1, n = 170, test scores : [0.925 , 0.96666, 0.975], avg : 0.95555
k = 3, n = 170, test scores : [0.85833, 0.875. , 0.86666], avg : 0.86666
k = 5, n = 170, test scores : [0.775 , 0.80833, 0.8], avg : 0.79444

(test scores is got by validation set)

從cross validation的結果可以發現, k = 1, n = 170的時候average test scores最高, 因此我會選擇k = 1, n = 170作為hyper parameter

2-5

Predict : [1 2 3 4 40 6 7 8 9 38 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40]
Labels : [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40]
Recognition ratio = 38/40 = 0.95

3-1

使用此filter的效果可以降低圖片中細節的資訊, 通常拿來去除雜訊, 但使用的同時會將圖片中的邊緣模糊/平滑化

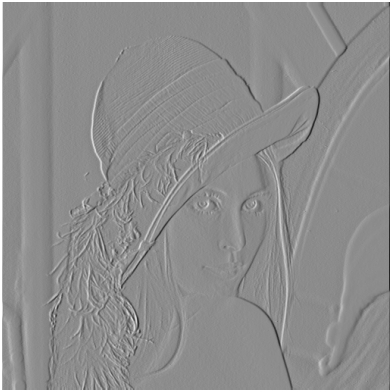

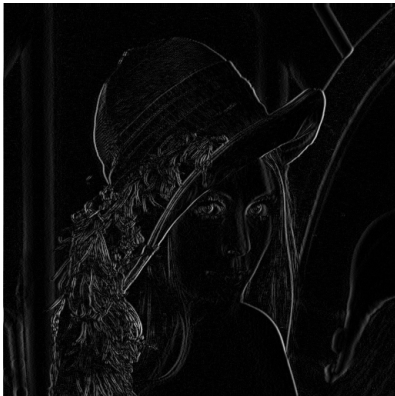
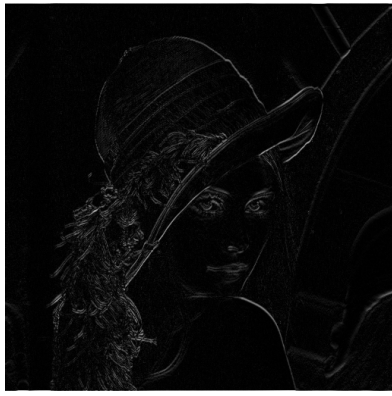
Result :





3-2

$$k_x = [0.5, 0, -0.5]$$

$$k_y = [0.5, 0, -0.5]^T$$

	lx	ly
Normalized		
Normalized and absolute		

	Lena	Gaussian-filter
images		

這邊我稱先取gaussian-filter後的magnitude稱為gaussian magnitude image, 而直接對lena取magnitude的結果稱為magnitude image.

可以發現gaussian magnitude image在圖像邊緣的強度上並沒有magnitude image. 來的高, 以及圖像的細緻度上gaussian magnitude image也沒有magnitude image來的高, 例如 : magnitude image中lena的帽子花紋保存得比較完整

我認為主要原因是 : 經過gaussian filter後的圖像都會損失掉部分的細節, 而edge(ex : I_x , I_y)也是細節的一部分, 因此在算magnitude的結果, magnitude image會比gaussian magnitude image來的細緻。