Computer Vision HW3 Report

101060003 曾守曜

I. K-means clustering image segmentation

結果中**左邊**的圖是用 kmeans++的方式初始化，**右邊**的圖是用自己選的點做初始化。

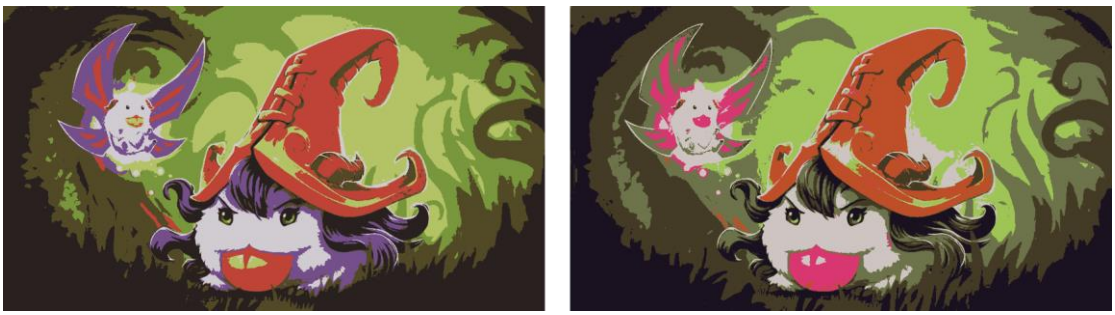
Manually selected points

K=3 K=7 K=11

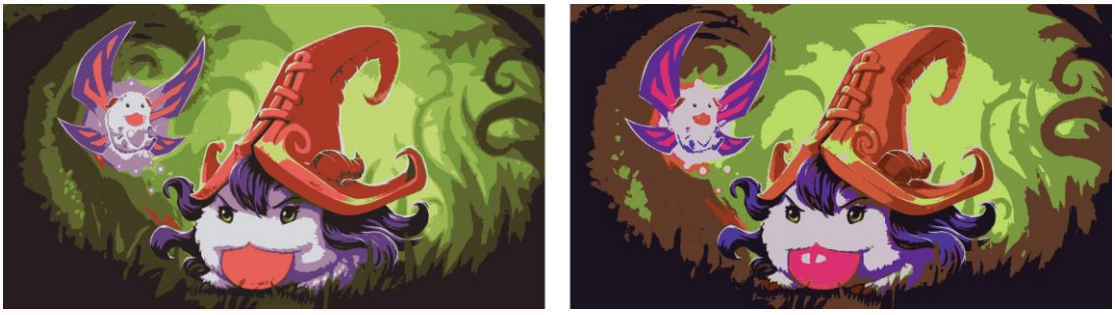
a. RGB space

Results of K = 3

Results of K = 7

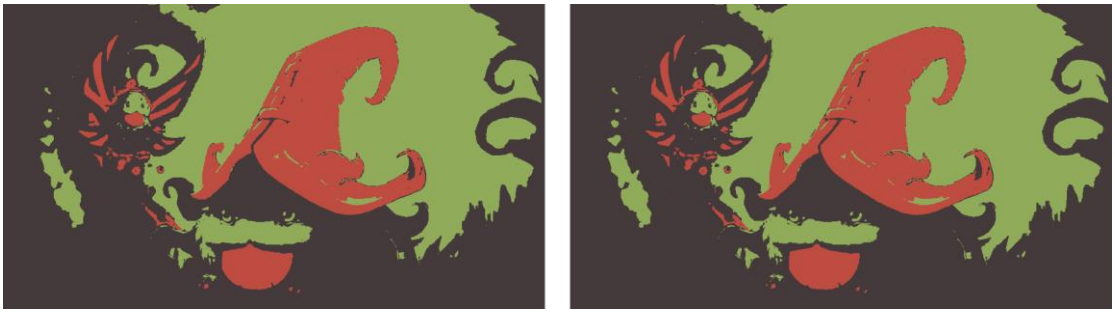


Results of K = 11

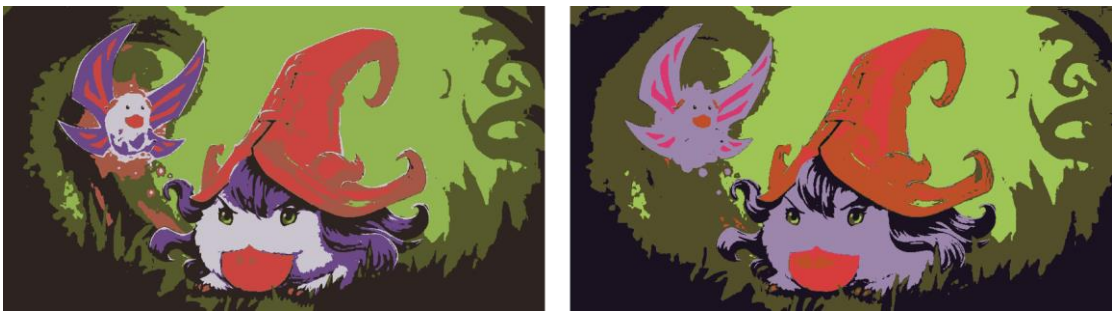


b. Luv space

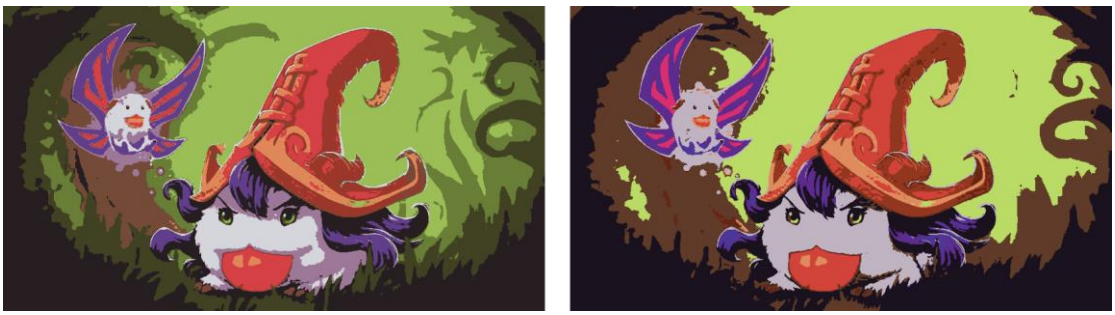
Results of K = 3



Results of K = 7



Results of K = 11



c. Discussions

不管是在 RGB space 或是 Luv space 中，當 cluster 的數目 K 很小時 (e.g. K=3)，kmeans++ random initialization 與 manual initialization 的結果沒有任何差異的可能性很大，因為 cluster 的數量太小，所以不同初始化的結果都會收斂到相同的 cluster centers。

當 K 提高時，就可以明顯看出兩種不同初始化方式會收斂到不同的結果，就整體的效果來看，我覺得 kmeans++可以收斂到不錯的結果，比起用 manual 方式來得更好，即使我在選擇初始化的點時，有刻意選擇人眼看起來不同的顏色，但是在 K=7 的比較中，manual 的結果有明顯色彩偏差，不盡理想。在 K=11 時，色彩偏差的情況變得比較輕微，不過 kmeans++的色彩還是比較好，然後在圖案細節部分還是由 kmeans++明顯勝出。

比較 RGB 和 Luv 的結果發現，Luv 會偏向把色彩相近但亮度不同的

顏色，分成一群(距離比較短)，從圖中背景漸層的綠色可以看出這個現象，推測是因為 Luv 把顏色分成亮度(L)與彩度(uv)兩部分，使得在計算差異時 uv 的值會 dominate。所以在 Luv space 中，K 要夠大才能保留足夠的細節，多大就需要看圖片色彩繽紛的程度。

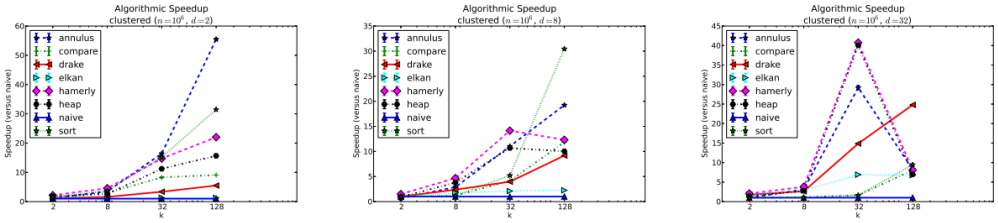
d. Implementation

1. 首先我使用 kmeans++來作 initialization，使用 randsample()來達到 weighted random sampling 的功能。
2. 在計算某個 data 點 x 和某個 center 點 c 的距離時，我使用下面的式子來作簡化：

$$\|x-c\|^2 = \langle x-c, x-c\rangle = \langle x,x\rangle - 2\langle x,c\rangle + \langle c,c\rangle$$

   但是這個的簡化對 high-dimensional sparse data 才比較有用，以這次作業的維度來說，速度上沒有什麼明顯的提升。
3. 我的終止條件設定是，當 data points 的 label 沒有再變化時就結束。

RGB image data 只有 3 維，算是相當低維度的 data，我有調查過許多 k-means 的加速演算法，根據下面這張圖表，hamerly(2010)的演算法（粉紅色），在低 data 維度（d）和低 cluster 數目時，有不錯的加速效果相較於 naïve 的 Lloyd's algorithm。此外，hamerly(2010)是基於 Lloyd 去改進的演算法，如果想要進一步加快這題作業的運算速度的話，或許可以選擇實作 hamerly(2010)來加速 k-means clustering。



$d = 2$ $d = 8$ $d = 32$

- 50 true Gaussians, $n = 10^6$.
- K varies from 2 to 128.

BAYLOR UNIVERSITY

II. Background Replacement using k-means

因為這次要區分的背景與前景差異相當明顯，背景顏色也相當的單純，我選擇 K=2 來做 k-means clustering。

再來為了加速處理的速度，我只對影片的第一張 frame 作 k-means clustering，因為影片的內容物件都是相同的，色彩也沒有太大的改變，所以這樣的方式是可行的。之後的 frames，就用第一張所計算出的 cluster centers 作 labeling，區分出前景和背景。因為每張 frame 的最左上角都是背景，所以我總是取 data 中第一個點的 label 當作背景的 label。

用上述的方式做完 segmentation 後，前景的豹的周圍還會有點殘留背景的紫光，所以我會再用 3x3 的 square element 做 erosion 把邊緣吃掉一點點。為了讓貼上新背景的影像結合得更柔和，我會對前景的邊緣做 Gaussian blur，使得邊緣附近的值會結合前景與背景的色彩，而非單單從一邊取值。實際的做法是對前景的 binary mask 做 Gaussian blur，前景內部都是 1 的部分會得到相同的值，只有 0,1 交界的邊緣部分會有數值上的差異。

III. Mean-Shift image segmentation

a. Mean-shift on color space only

在沒有限制 spatial bandwidth 的情況下，mean-shift 的計算量會很大，造成速度緩慢，所以我有實作助教提供的 **early stop** 和**桃太郎**加速法。但即使如此，計算時間還是很久，所以我做了**第三種加速**，那就是在每次 mean-shift 收斂的時候，除了確定原本的中心點為 mean 值之外，還會將在 color bandwidth 內的所有點，都指定成該 mean 值，這些已經被指定過數值的點，就不會跑 mean-shift algorithm。

結合以上 3 種加速方式，就可以在 1 分鐘或是數分鐘內跑出這次作業的 segmentation 結果，時間受 color bandwidth h_r 影響。從下面的結果可以看出，雖然做了不少的簡化，但是仍然有不錯的效果。

第三種加速方法的結果有一個缺點，就是收斂的結果會受處理 data points 的先後順序影響，如果希望盡量避免這層影響，可以對 data 作 random permutation，然後多做幾次 mean-shift，選擇較佳的 segmentation 結果。結果的好壞或許可以用 $\frac{intra-class\ similarity}{inter-class\ similarity}$ 來衡量。

RGB space, $h_r = 16$

Luv space, $h_r = 16$

early stop 的 threshold value 是一個介於 0~1 之間的值，相同的值可以適用於 RGB 和 Luv 兩個 color space，因為在與 threshold 比較時我會對 meanshift 的值除上一個 normFactor：

1. RGB: 我假定輸入的 RGB 影像值是介於 0~255，
   所以 normFactor = 255。
2. Luv: u,v 的值有負有正且沒有明確的界限，L 會介於 0~100，所以我取 normFactor = max(abs(all data scalar values))

b. Mean-shift on color and spatial space

一開始我先實作的其實是有 color bandwidth 和 spatial bandwidth 的 mean-shift algorithm，~~因為在網路上先查到這種~~。當 spatial bandwidth 不會太大時，適當的調整 threshold，就可以在不會太久的時間得出還可以的結果，所以這部分的演算法我還沒有使用桃太郎和第三種方式加速。

如果得要到比較漂亮（明顯的）segmentation 結果，threshold 還是

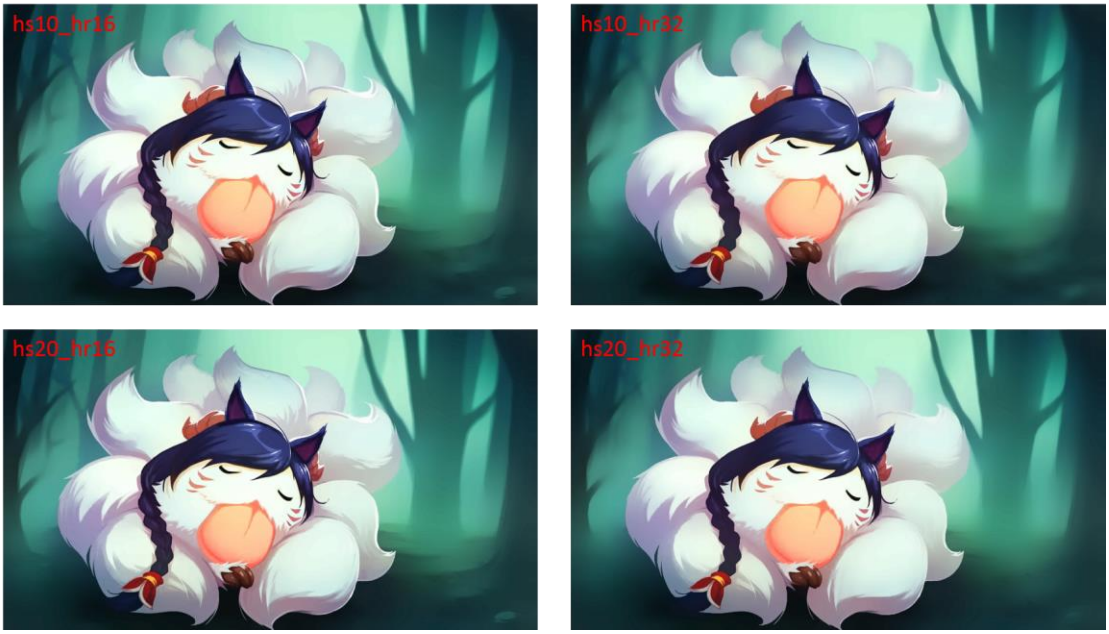
必須調低一點才行，不過這樣算一次的時間可能需要約 1~2 小時不等，spatial bandwidth 調愈大就愈久，調到跟原圖一樣大就近似於只有 color information 的 mean-shift 了。

c. h_s and h_r for RGB and Luv

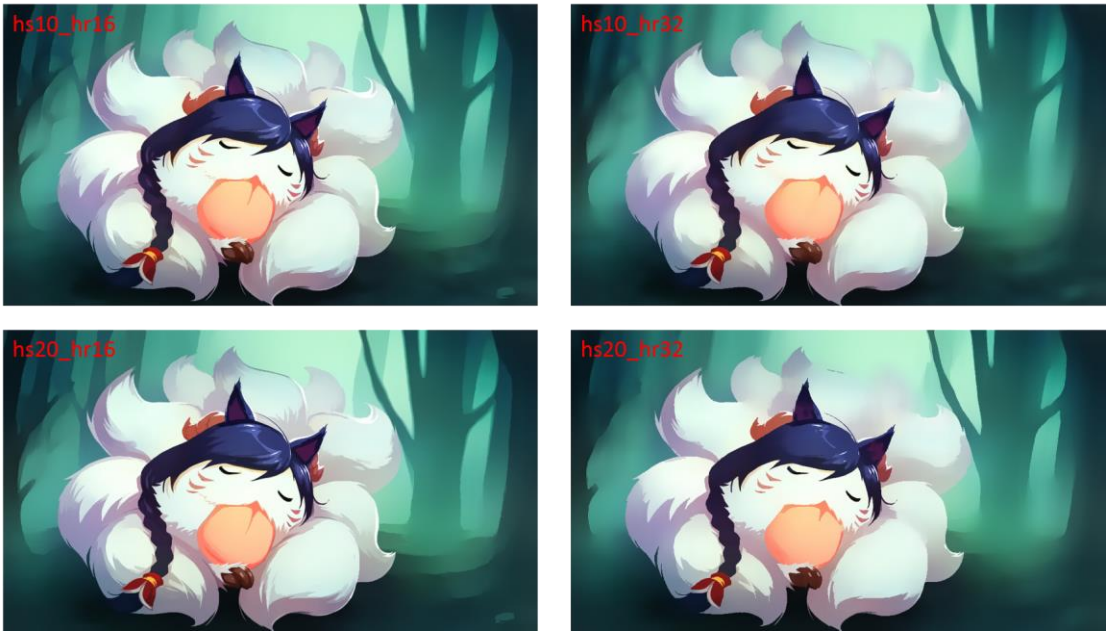
1. h_s 與 color space 無關，所以我採用相同的兩個值：10、20。
2. h_r 就與 color space 有關了，我以人眼的方是判斷可以 segment 出相近效果的兩組值：RGB = (16, 32), Luv = (7, 14)。

d. Comparing color-space mean-shift segmentation on RGB and Luv

Luv color + spatial info



RGB color + spatial info



從上面的結果可以觀察出，當 color bandwidth 愈大，segmentation 結果愈 coarse，色塊愈大塊、邊界也愈模糊。比較特別的是，在 spatial bandwidth 都不大且是 uniform kernel 情況下，spatial bandwidth 較大，反而會產生邊界比較清楚的 segmentation 結果，應該是因為觀察的範圍比較廣，所以可以收斂到更分離的位置，而不會因 spatial bandwidth 太小、觀察的範圍小，使得 mean-shift 會收斂在許多比較局部的位置上。