DIP FINAL PORJECT

Exploiting Perceptual Anchoring for Color Image Enhancement

第 12 組 黃湛元

黃煜堯

曾勁凱

教授：陳宏銘 教授

O u t l i n e

- **Exploiting Perceptual Anchoring for Color Image Enhancement**
- **Gamma Correction – CIELUV & HSV & YCrCb & RGB**
- **Model Analysis - CIELAB & HSV**
- **HSV Enhancement**
- **Low Light Enhancement**
- **Color Correction**
- **Color Correction + SLIC + CIECAM02**

Exploiting Perceptual Anchoring for Color Image Enhancement

根據 reference paper“Exploiting Perceptual Anchoring for Color Image Enhancement”以及 slide 上面的步驟，基本上就可以一步一步實踐這個方法。

1. 讀取 image，此時因為用的是 opencv，因此會是 BGR 格式，先將其轉換為 RGB 方便之後的計算，並且將其除以 255，讓 range of value = [0, 1]。

2. 根據 slide 設定參數，將 γ_{rf}、γ_{gf}、γ_{bf}、M_f、γ_{rl}、γ_{gl}、γ_{bl}、M_l 設定好，接著透過公式將 image 從 RGB-color space 利用 γ_{rf}、γ_{gf}、γ_{bf}、M_f 轉換成 XYZ-color space。另外，設定一組 white color [1, 1, 1]，也將其轉換成 XYZ-color space。

3. 然後 Y_b、L_a、c、N_c、F，加上前面計算出來的 image XZY 以及 white XYZ 一起丟入 CIECAM02 model 之中，計算 hue、lightness、chroma，計算方法如 slide 所述。

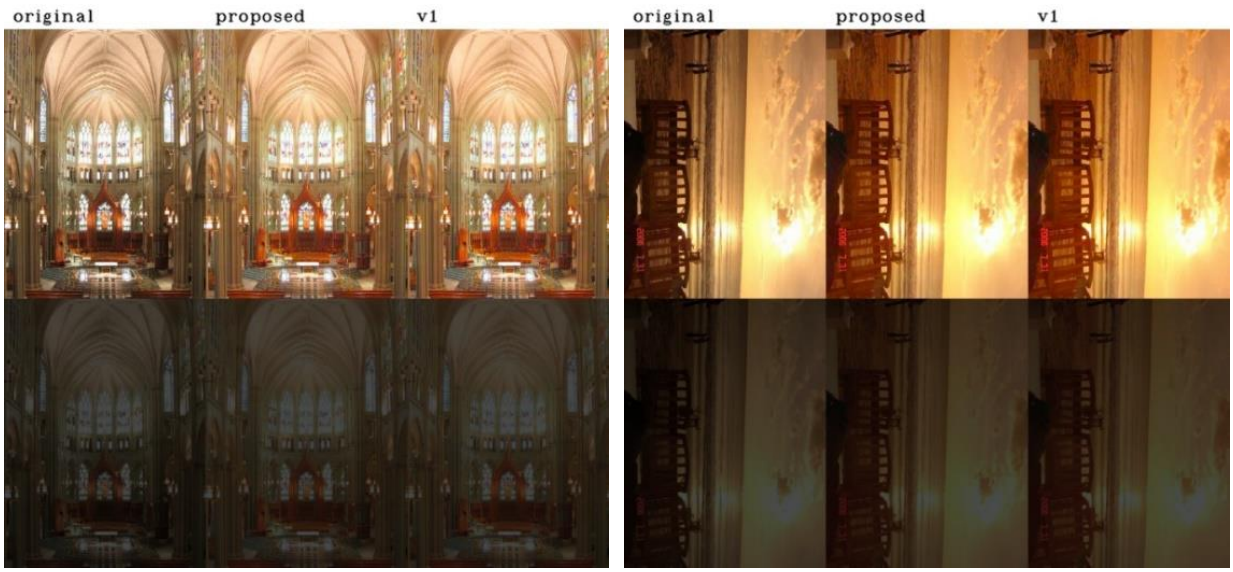
   此時所用到的 CIECAM02 model，我們是根據 python library 之中的 colormath 這個 library 的 source code 去做改寫，放在 ciecam.py 之中。

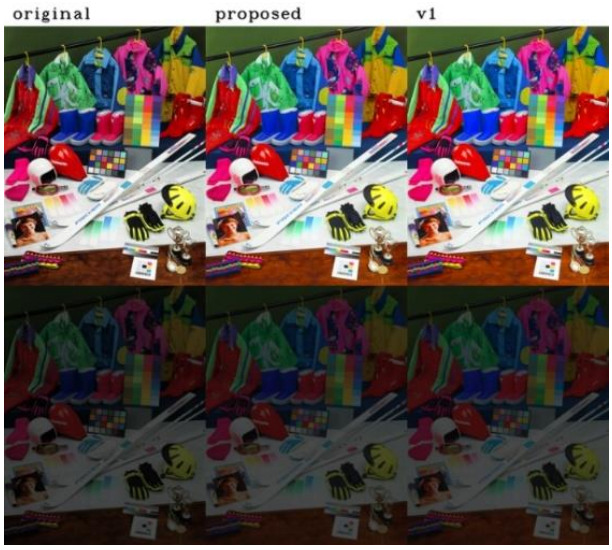
4. 在這個步驟需要將 CIECAM02 model 反向的做一次，但是必須先將 white XYZ 改成利用 γ_{rl}、γ_{gl}、γ_{bl}、M_l 所轉換成的 white XYZ。Model 的公式則可以從 slide 之中，將原本的等式做 inverse，即可計算，最終得到 enhanced XYZ。

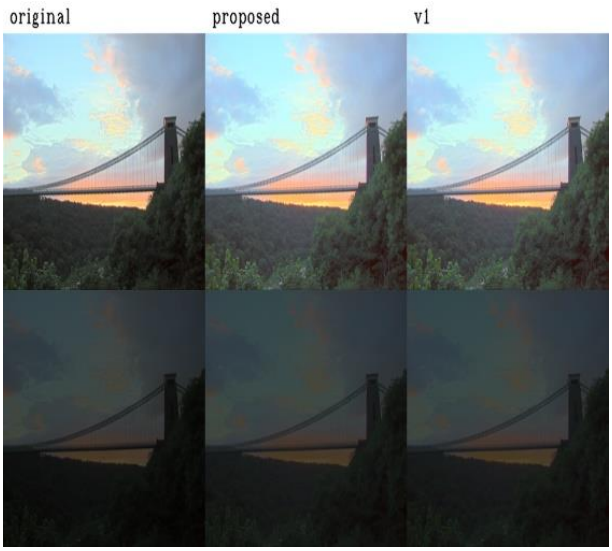
5. 最後繼續透過公式將其轉換成 RGB-color space，然而此時會發現因為 enhancement 的關係，會使其 range of value = [0, ~]，因此需要透過 clip function 來進行裁減。

   而經過 clip 之後，在較亮 or 白的部分，會出現 loss of details 的情況，因此透過公式配合 lightness J & chroma C 來做調整，使其變成一個 curve，可以 one-to-one 的 mapping。

實作的結果如下圖，最左側為原始圖片，中間為 proposed results，右側則為我們實作的結果。







程式部分，ColorPatchEnhanced.py 是使用資料夾 images/color patch 之中的圖片去做論文的實作，而 NaturalImgEnhanced.py 則是使用資料夾 images/natural image 之中的圖片去做實作。對應的結果會存在 images/color patch results 以及 images/natural image results 之中，取名為 XX_v1.png。

另外，相同的 RGB 值會轉換成相同的 enhanced RGB 值，因此可以利用 dynamic programming 的技巧來減少 computation cost，將計算過的 RGB 值對應的 enhanced RGB 值存起來。又或者可以存全部的可能性，讓整個過程便成 table look up 的概念。

● Gamma Correction – CIELUV & HSV & YCrCb & RGB

由於在 paper 中的效果比較，有提到使用 gamma correction 的結果，但只有一個形式的結果，因此想嘗試透過針對不同的色彩模型特定 channel 去做 gamma correction 並比較一下在不同色彩模型下的結果。Gamma 值設定為 0.6。

這次的必要主要是針對這幾個色彩模型去做比較。

	RGB	HSV	CIELUV	YCrCb
Channel	R、G、B	H、S、V	L、U、V	Y、Cr、Cb

其中 HSV、LUV、YCrCb 中 H、L、Y channel 都是代表亮度的部分，其他 channel 則是在色彩上有不同的表示。針對亮度部份的 channel 比較，可以看到說在不同的色彩模型下同樣針對亮度去做調整會有不同的結果。

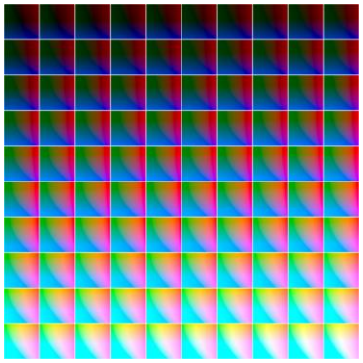
可以看到雖然結果的差異不大，可以看到在兩張結果上都是 HSV 上表現比較好，顏色的部分在肉眼觀察下比較接近原圖的色彩，在第一張圖夕陽的部分也保留比較好的細節，在天空的細節上則比較沒有明顯的差異。第二張圖則可以看到在相比之下 HSV 也是有比較好保留原本的細節。但兩張 sample image 在調整過後，low back light 的情況下其實沒有好的 enhancement 的效果。

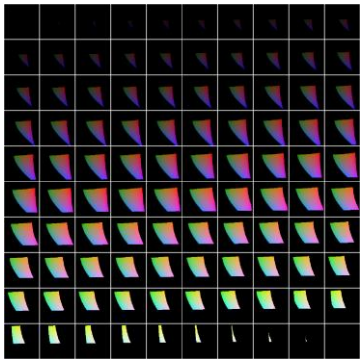
程式部分 gamma_test.py 是分成 4 個不同 function，分別去讀取 test_image / image 的原圖去做調整，再透過 SimulatingImgDimBack.py 去轉成 low back light 的形式。

- ## Model Analysis - CIELAB & HSV

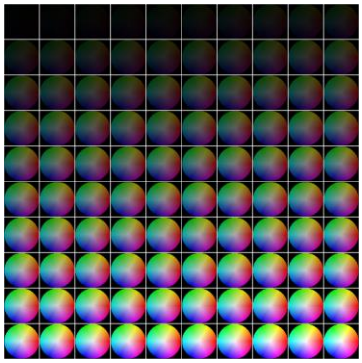
參考資料所使用的 CIECAT02 色彩空間在計算上雖然需要六項視覺屬性，但實際上改動的主要參數只有 Lightness(明度)、Chroma(彩度)及 Hue(色調)。由於本學期並沒有教到 CAT02 如此複雜的模型，另外在課程中所提到的 HSV 及 CIELAB 色彩空間也都具備著此三項與裝置無關的色彩屬性，照理也應該能達到類似的成果，因此我們在這次報告中嘗試將本學期所學到的方法與上述兩項色彩模型結合，利用較簡易的方式進行影像強化。

根據我們的理解，題目的要求除了增強彩色影像以外，更重要的是增強效果不能隨環境亮度的改變而有所折扣，換句話說就是要在不改變影像亮度上下限的前提進行彩度最大化。因此，首要之舉便是觀察 HSV 及 CIELAB 色彩模型在不同亮度下所能提供的色彩範圍。





上圖分別為 CIELAB 色彩模型在 L=1~100 中對應的顏色(左)以及經過 sRGB 的 gamut mapping 後實際能輸出的顏色(右)，切片的 x 軸為調整紅/綠的 a*，y 軸為調整黃/藍的 b*。經過 gamut mapping 後可以發現，CIELAB 色彩模型在不同 L 值下的 a*b*都無法以簡單的線性關係呈現。相較之下，HSV 色彩模型的 S 值及 H 值分別為半徑與角度，因此在不同 V 值下的色彩範圍(下圖)皆能以圓形分布。經過比較後，下一節的影像強化將會使用 HSV 系統進行調整。

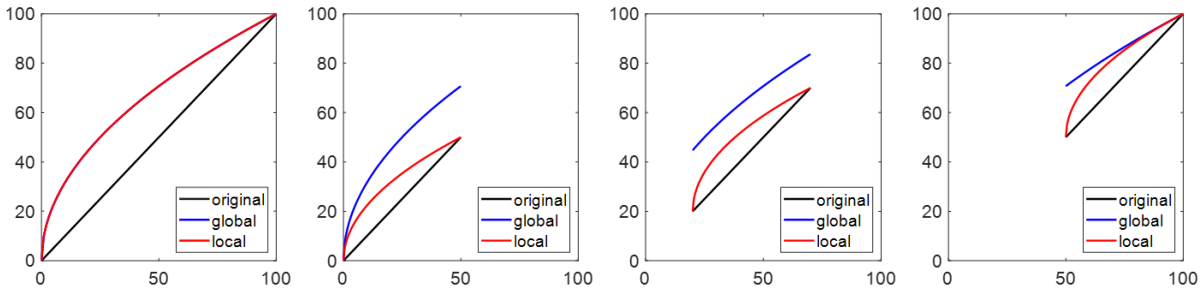


● HSV Enhancement

1. 強度(Value, V)調整

從 HSV 模型不同 V 值的色彩分布中可以發現，光是調整 V 值就能極大影響視覺的色彩辨識度以及後續兩項參數的操作空間，因此第一步便是對影像對比度進行增強。但在上一節中，我們提到題目的核心限制是不改變影像亮度上下限，因此在 gamma 校正前須先對整個影像進行正規化，校正後再將值域調整回原先的定義域中。下圖分別顯示了不同 V 值範圍在正規化前後進行 gamma 校正的差異，本報告使用之 gamma 值為 0.5。

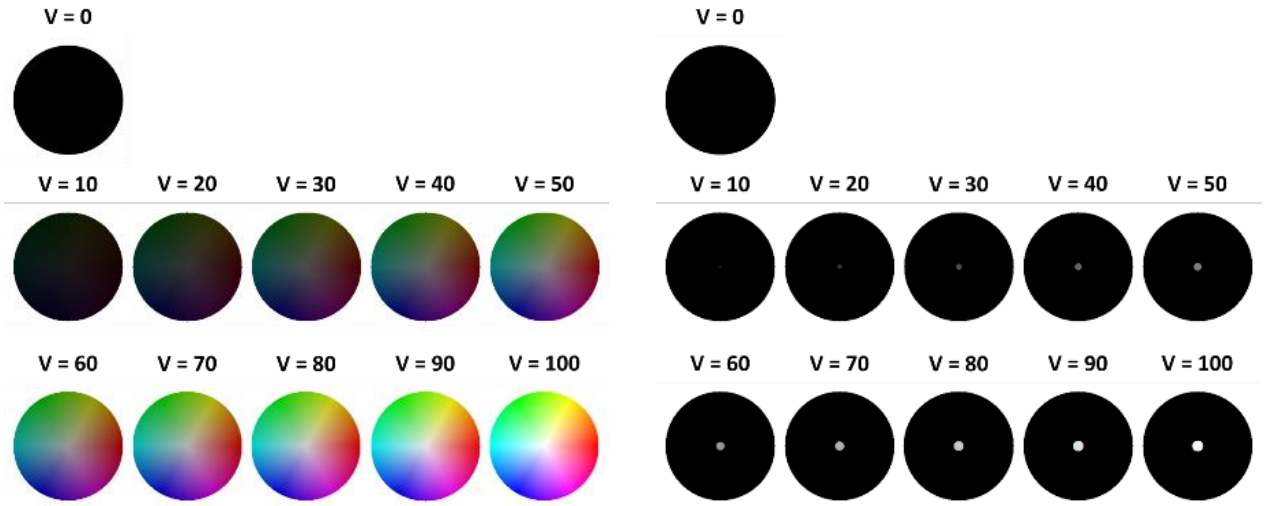
全域 | **區域**



2. 飽和度(Saturation, S)調整

在 HSV 色彩模型中，S 值越大越能看出色調，S 值越小則越接近灰階分布(如下圖所示)，也就是說提高影像 S 值便能保留彩度，因此我們對 S 值也進行了冪次為 0.5 的 gamma 校正。如同第一步的做法，在對 S 值進行 gamma 校正前也須事先正規化，並在校正後將值域轉回定義域，以避免因為原先影像彩度範圍過小導致校正後反而出現褪色或過度彩化的現象。

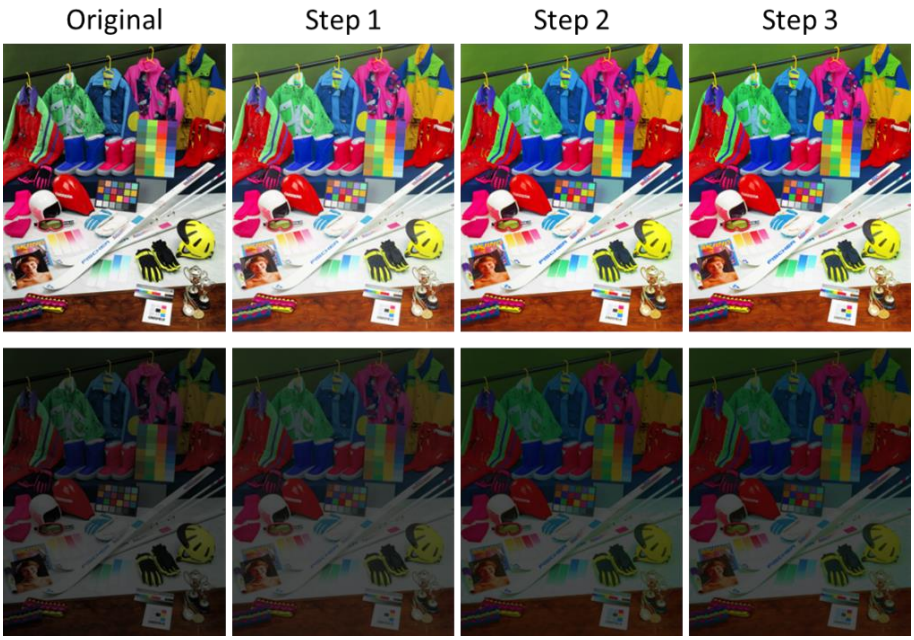
在 HSV 色彩模型中，控制色調的 H 值並沒有黑白兩色，使得即便事先對彩度正規化再進行 gamma 校正，也會因為灰階區域受到的校正幅度較大，導致黑灰白等色調仍然有過度彩化的跑色問題。針對這個問題，我們決定了每個 V 值所可接受的灰值區域，涵蓋在區域內的色塊將會在校正後回復成其原始的飽和度。下圖分別為 HSV 色彩模型在不同 V 值的色彩分布(左)以及其對應的可接受灰值區域(右)。



3. 色調(Hue, H)調整

如同在上一步驟中，我們不希望灰階區域在濾鏡的作用下出現跑色的問題，因此色調在影像強化的重要性遠小於其他兩樣。但適度地調整色調能降低色調過度集中的問題，進而平衡影像整體的色彩分布，因此我們最後一步便是將色調切成不同等分，並對各區域的色調進行線性正規化。在 HSV 色彩模型中，色調大致可分為紅-黃-綠-藍-靛-紫，但為了避免線性正規化造成色調失準，因此我們額外細分成 12 等分(如下圖所示)。

依序進行上述三個步驟所呈現的結果如下所示。在第一步的強度調整後可以看出不論是原始影像或是經過暗化後的影像，其亮度都有顯著的提升。但由於亮度會在一定程度上影響色彩對比度，看起來反而便淡了，因此在第二步的飽和度調整後，便能感受到明顯的對比差異。而最後的色調調整上雖然看上去幾乎沒有變化，但實際上運算的數值結果是有更動的，因此在考量各方面因素後仍然決定保留此步驟。



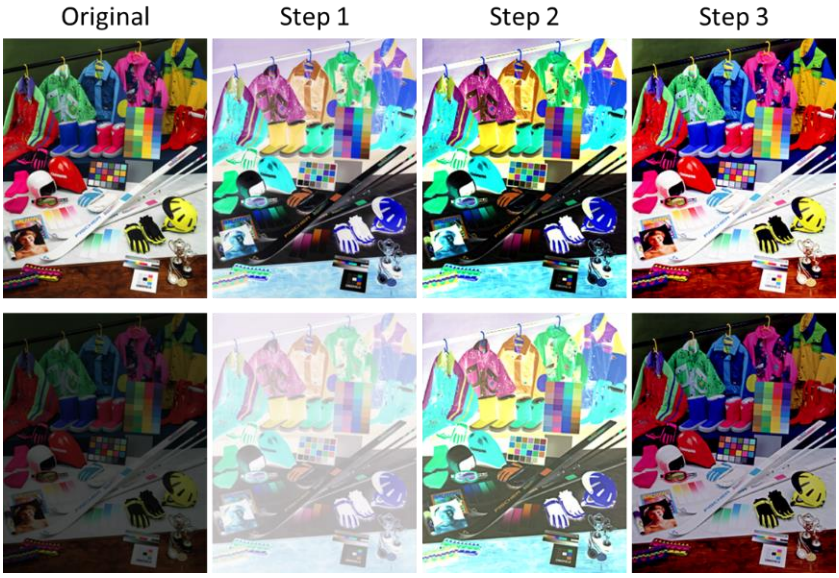
- **Low Light Enhancement**

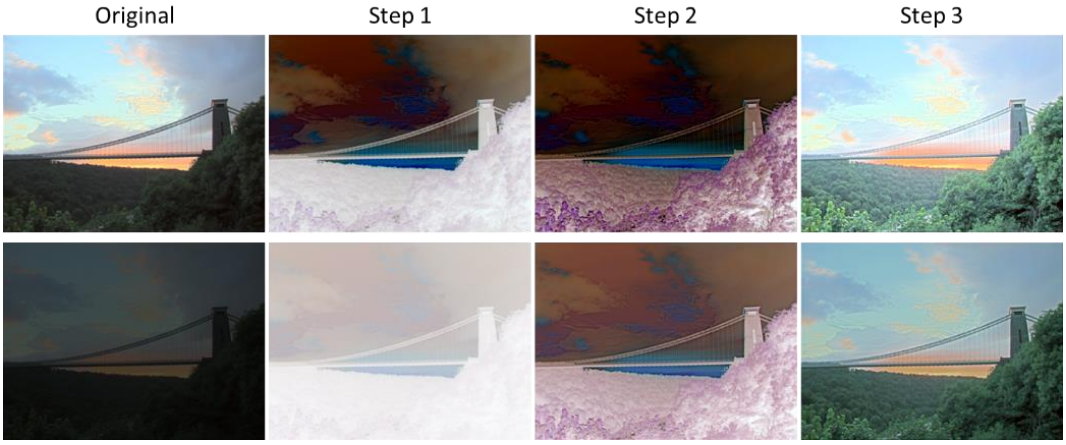
Xuan Dong 等人在 2011 年發表了一篇論文 - 《Fast efficient algorithm for enhancement of low lighting video》，其內容是針對極低光照影像進行影像增強。雖然這項方法並沒有滿足本次題目中彩度最大化的要求，但就如前面所提到的，僅是改善亮度也能大幅影響視覺上的色彩辨識度；另外，其中的計算關係雖然複雜，但在 MATLAB 當中僅須三行即可完成處理，執行起來非常容易，因此我們決定將此方法與我們的 HSV 系統增強方法進行比較。

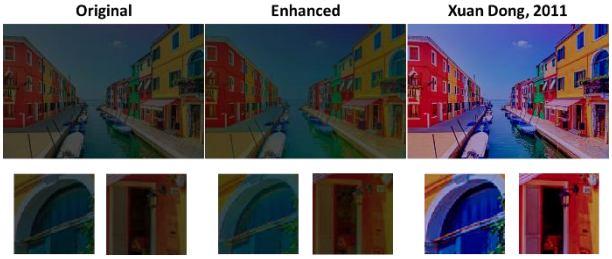
和有霧影像相同，低光照影像中的背景在進行負片濾鏡後的亮度值都很高，但近物的 RGB 通道中至少會有一個極低值，因此作者提出直接用除霧的運算方法對低光照影像的負片影像進行處理後，再取一次負片效果即可獲得增強的影像。其步驟如下：

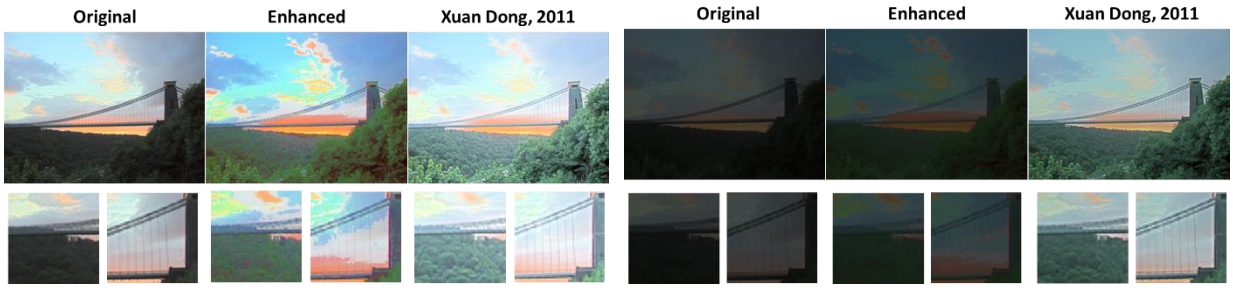
1. 利用函式 imcomplement()反轉影像取得負片效果
2. 利用函式 imreducehaze()對負片影像進行除霧處理
3. 利用函式 imcomplement()反轉除霧後的負片影像，即可取得增強影像

依序進行上述三個步驟所呈現的結果如下所示。



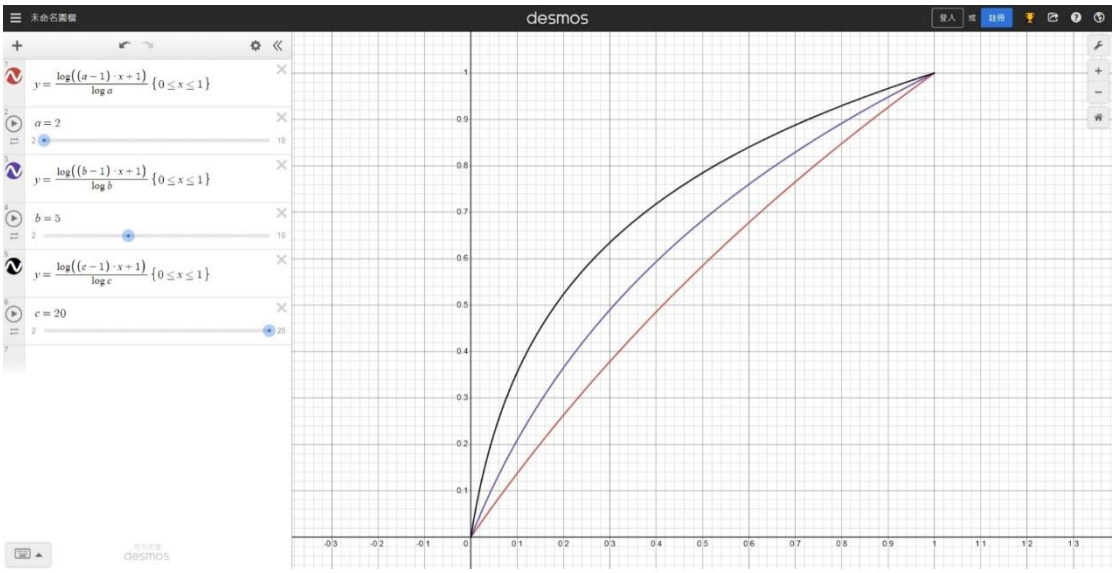




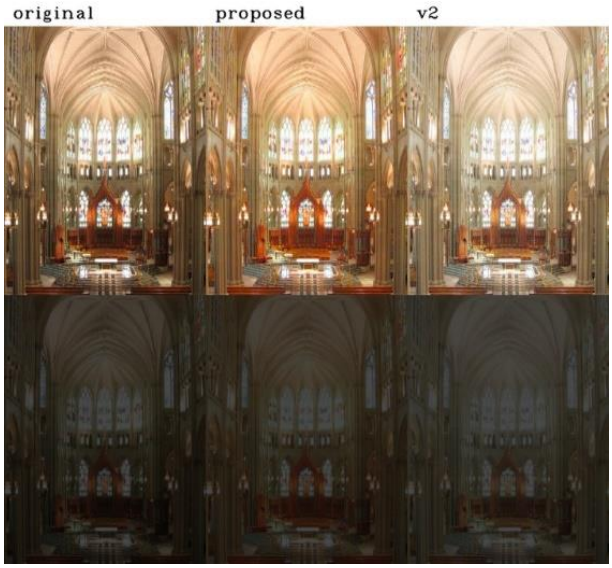


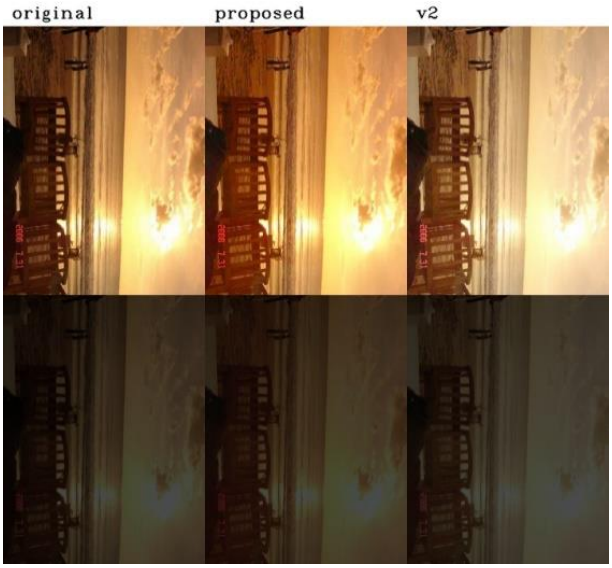
- **Color Correction**

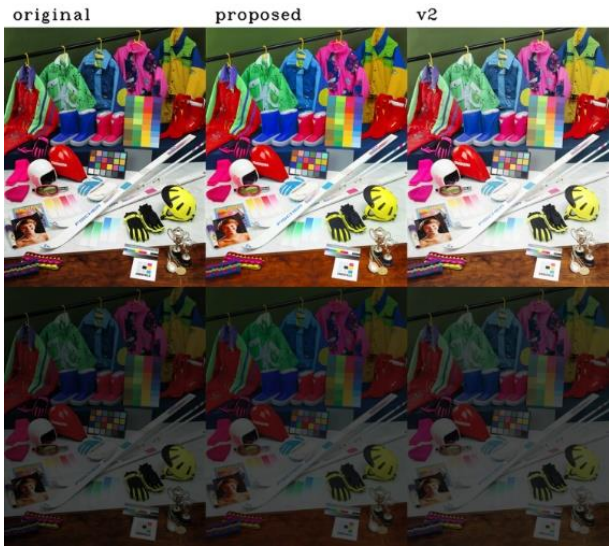
根據上課內容，可以知道 tone and color correction 也可以將圖片變亮，因此利用下面公式去做 enhancement：

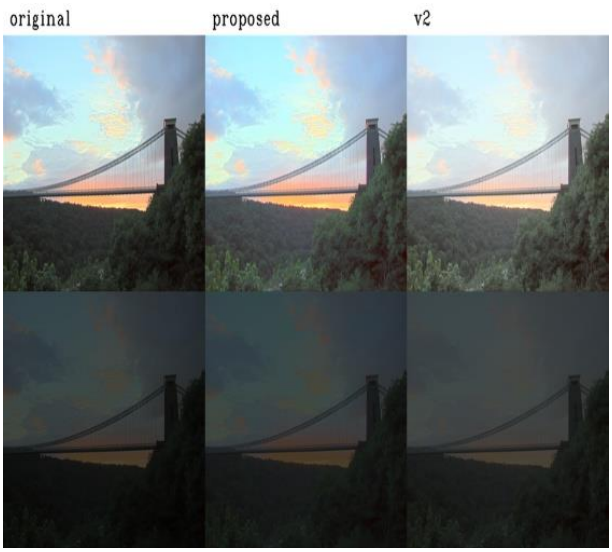


結果如下，最左側為原始圖片，中間為 reference paper 的結果，右側則為 tone & color correction 的結果。







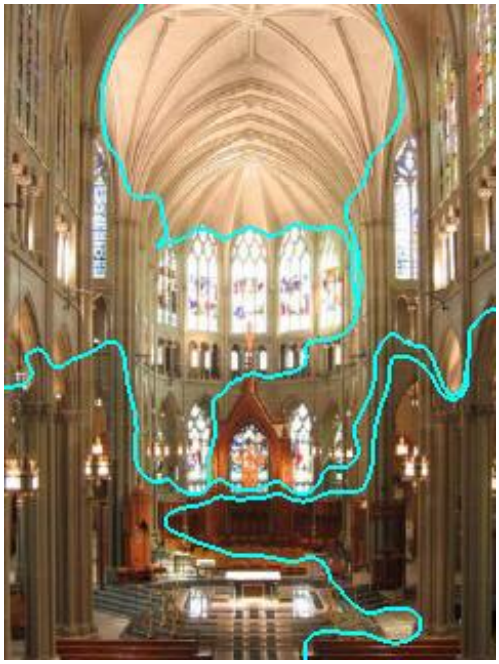


從圖片可以看出，整體是有 enhancement 的效果。但是因為目前參數 a 是定值，因此如果 a 太小的話，暗處的細節就無法獲得足夠的 enhancement；但是如果參數 a 太大的話，會使得較亮 or 白色的部分變得分不清差別。

程式部分，ToneAndColorCorrection.py 是使用資料夾 images/color patch 之中的圖片去做 tone and color correction。對應的結果會存在 images/natural image results 之中，取名為 XX_v2.png。另外，這個方法的運算速度很快，基本上只需要 0.1 秒就可以處理完一張圖片。

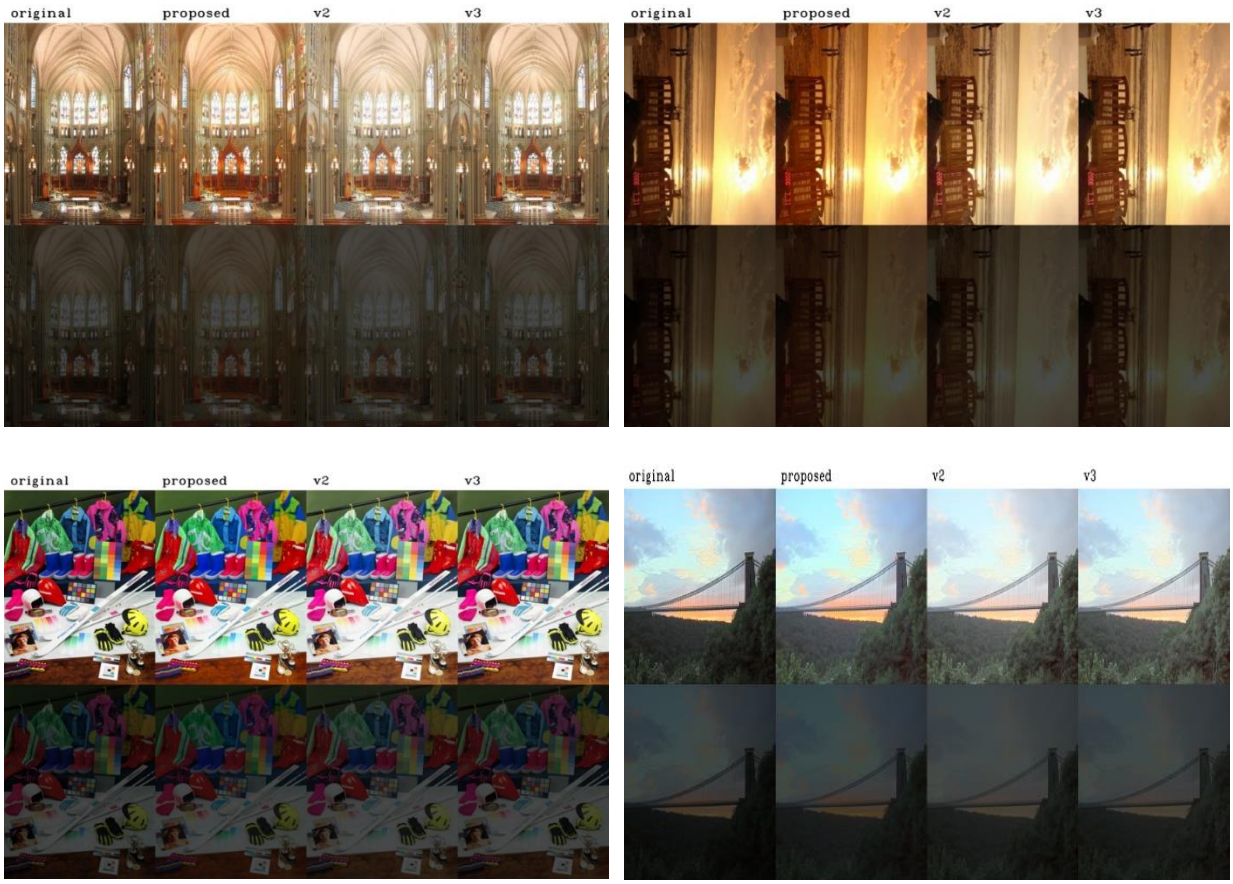
● Color Correction + SLIC + CIECAM02

由於前面的方法感覺是可行的，因此希望從此著手，利用 super-pixel 的概念，將 image 切割成幾個 groups，因為我們認為亮的區域會成一區，暗的區域會成一區，因此不需要分成太多區域，且同一區域可以用相同的參數去做調整。



另外，由於我們覺得可以代表一個區域的亮暗程度最方法，還是利用 CIECAM02 model 來計算 lightness & chroma 最方便，因此我們先計算 super-pixel 內的 mean color value，接著計算這個 mean color value 的 lightness J & chroma C。最後，由於我們希望亮的部份的參數 a 值要小，而暗的部分參數 a 值要大，因此設 a = sqrt(1 / (J*C))，這樣就能夠符合我們對於 a 的要求。

結果如下，最左側為原始圖片，中間為 reference paper 的結果，右側則為 tone & color correction + SLIC + 的結果。



從結果可以看出來，經過 SLIC + CIECAM02 的改善，有成功將亮的部分變得不那麼亮，同時保持暗的部分依舊有做到 enhancement。像是第一張圖片中上部分以及第二張圖的太陽部分，v3 的圖片比 v2 來講，亮度有稍微壓下來，但其餘部分卻依舊還是維持相同的亮度。而第四張圖，v3 的圖片右下角的樹林明顯比 v2 的圖片更亮，但 v3 圖片的天空反而比 v2 的圖片有些許的調暗。

程式部分，SLIC.py 是使用資料夾 images/color patch 之中的圖片去做 tone and color correction。對應的結果會存在 images/natural image results 之中，取名為 XX_v3.png。另外，這個方法的運算依舊速度很快，基本上只需要 0.3 秒就可以處理完一張圖片。