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

Outline

- 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_f、γ_f、γ_f、M_f、γ_l、γ_l、γ_l、M_l 設定好，接著透過公式將 image 從 RGB-color space 利用 γ_f、γ_f、γ_f、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。

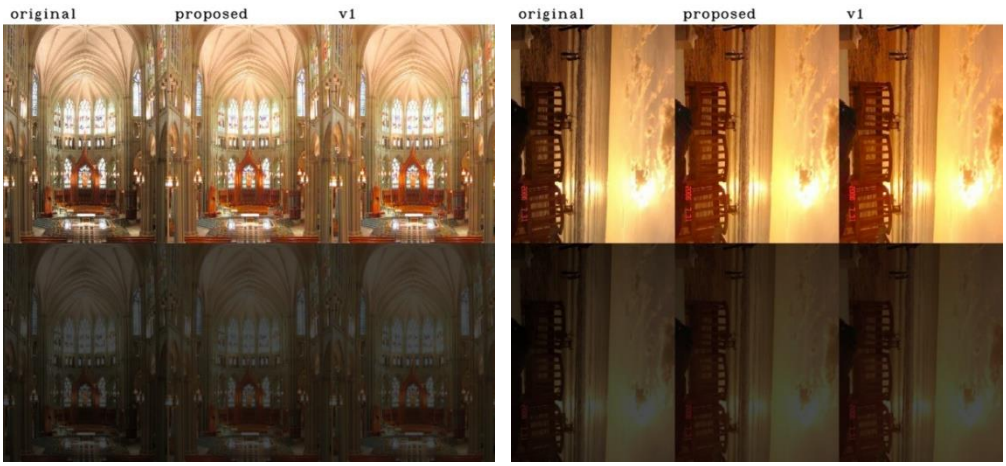
   此時所用到的 CIECAM02 model，我們是根據 python library 之中的 colormath 這個 library 的 source code 去做改寫，放在 ciecam.py 之中。

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_l、γ_l、γ_l、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**

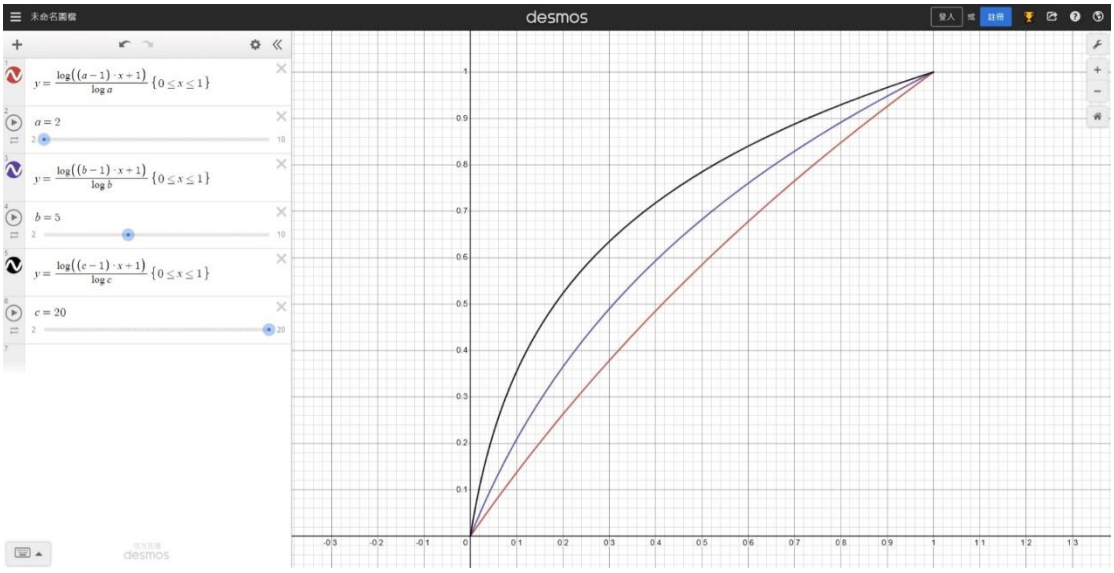
- **Model Analysis - CIELAB & HSV**

- **HSV Enhancement**

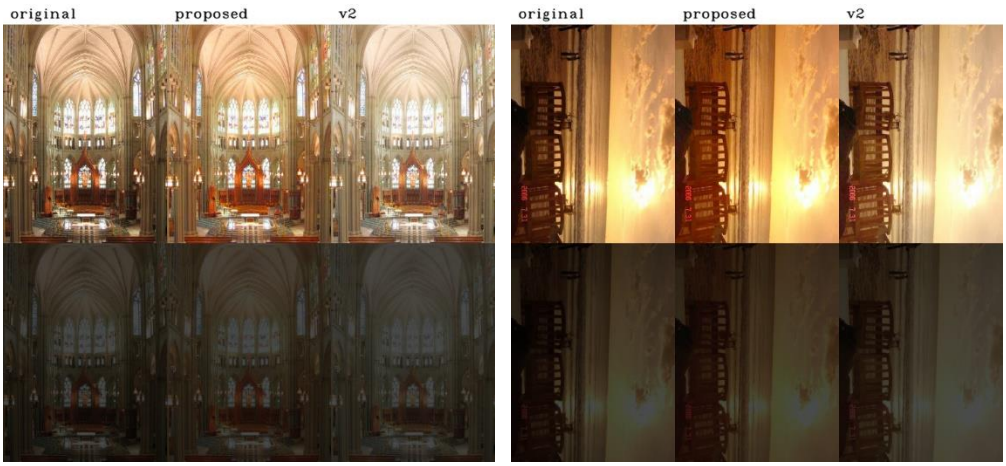
- **Low Light Enhancement**

- **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。

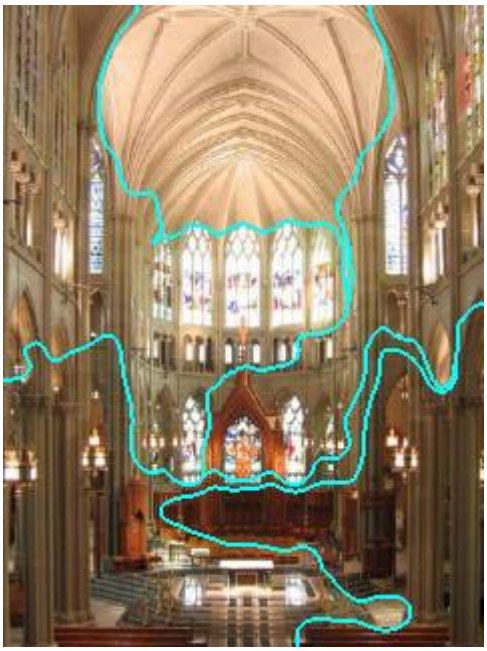


從圖片可以看出，整體是有 enhancement 的效果。但是因為目前參數 a 是訂值，因此如果 a 太小的話，暗處的細節就無法獲得足夠的 enhancement；但是如果參數 a 太大的話，會使得較亮 or 白色的部分變得分不清差別。

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。

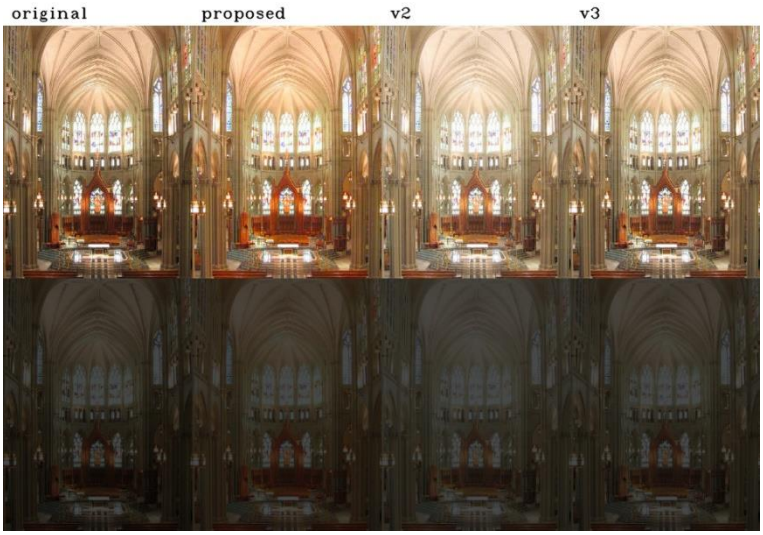
- **Color Correction + SLIC + CIECAM02**

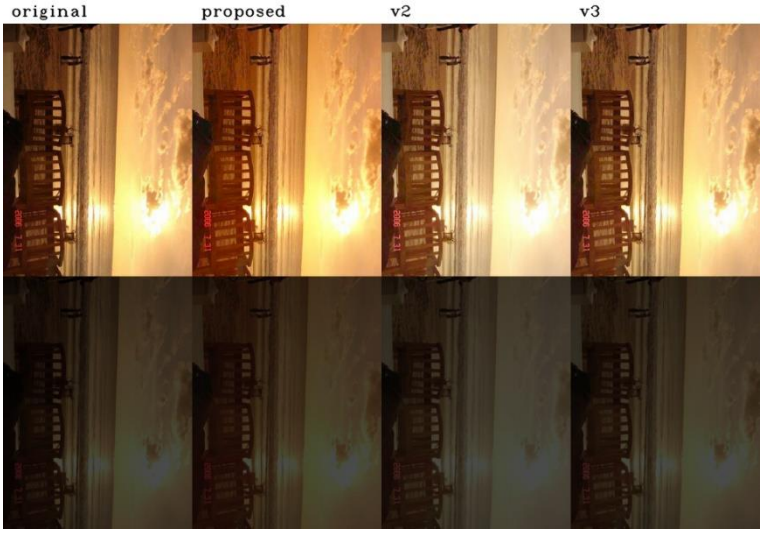
由於前面的方法感覺是可行的，因此希望從此著手，利用 super-pixel 的概念，將 image 切割成幾個 groups，因為我們認為亮的區域會成一區，暗的區域會成一區，因此不需要分成太多區域，且同一區域可以用相同的參數去做調整。



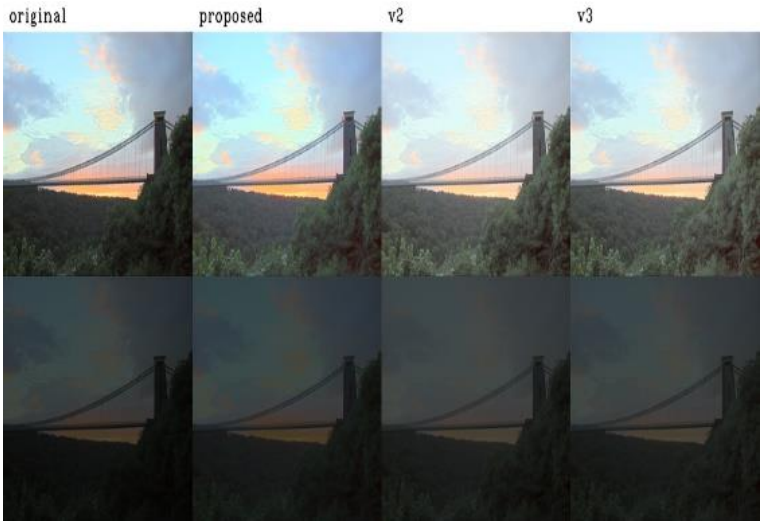

另外，由於我們覺得可以代表一個區域的亮暗程度最方法，還是利用 CIECAM02 model 來計算 lightness & chroma 最方便，因此我們先計算 super-pixel 內的 mean color value，接著計算這個 mean color value 的 lightness J & chroma C。最後，由於我們希望亮的部份的參數 a 值要小，而暗的部分參數 a 值要大，因此設 a = sqrt(1 / (J*C))，這樣就能夠符合我們對於 a 的要求。

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。


original
proposed
v2
v3


original
proposed
v2
v3


original
proposed
v2
v3



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。

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。