第12組 黃湛元

黃煜堯

曾勁凱

教授：陳宏銘 教授

DIP Final Porject

Exploiting Perceptual Anchoring for Color Image Enhancement

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。

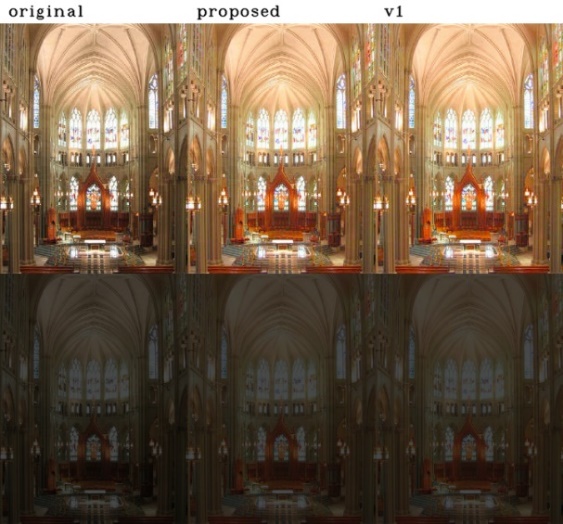
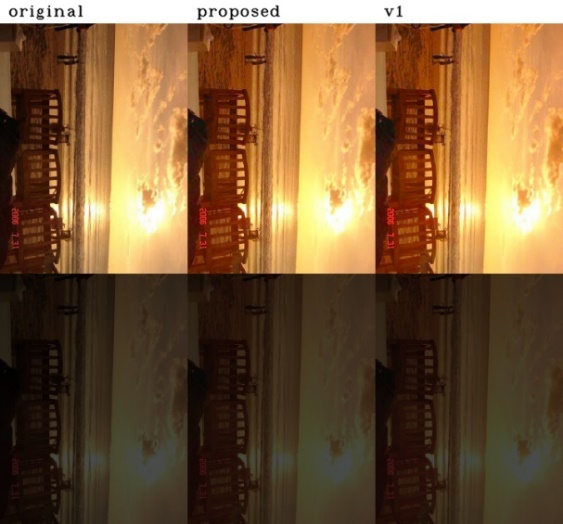
1. 讀取image，此時因為用的是opencv，因此會是BGR格式，先將其轉換為RGB方便之後的計算，並且將其除以255，讓range of value = [0, 1]。
2. 根據slide設定參數，將γf、γf、γf、Mf、γl、γl、γl、Ml設定好，接著透過公式將image從RGB-color space利用γf、γf、γf、Mf轉換成XYZ-color space。另外，設定一組white color [1, 1, 1]，也將其轉換成XYZ-color space。
3. 然後Yb、La、c、Nc、F，加上前面計算出來的image XZY以及white XYZ一起丟入CIECAM02 model之中，計算hue、lightness、chroma，計算方法如slide所述。

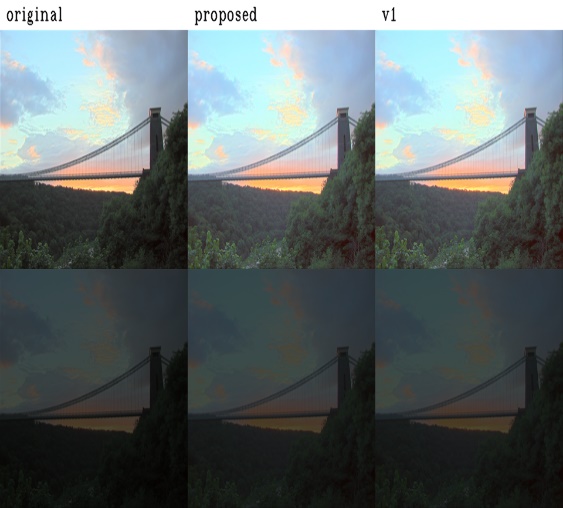
此時所用到的CIECAM02 model，我們是根據python library之中的colormath這個library的source code去做改寫，放在ciecam.py之中。

1. 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γl、γl、γl、Ml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。
2. 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。

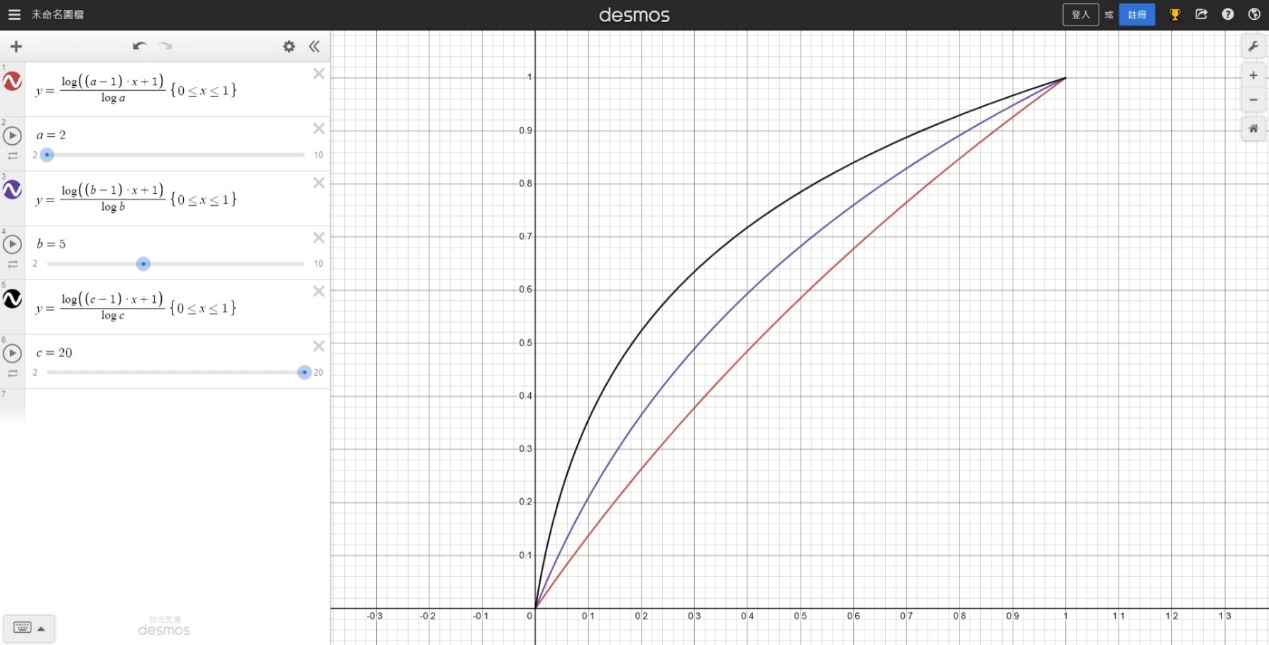
 

程式部分，ColorPatchEnhanced.py是使用資料夾images/color patch之中的圖片去做論文的實作，而NaturalImgEnhanced.py則是使用資料夾images/natural image之中的圖片去做實作。對應的結果會存在images/color patch results以及images/natural image results之中，取名為XX\_v1.png。

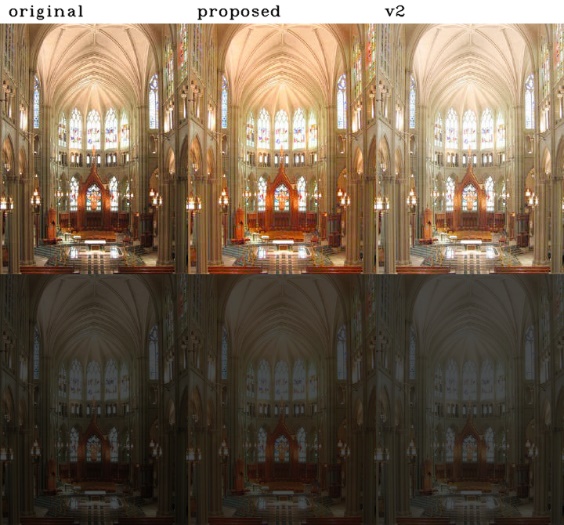
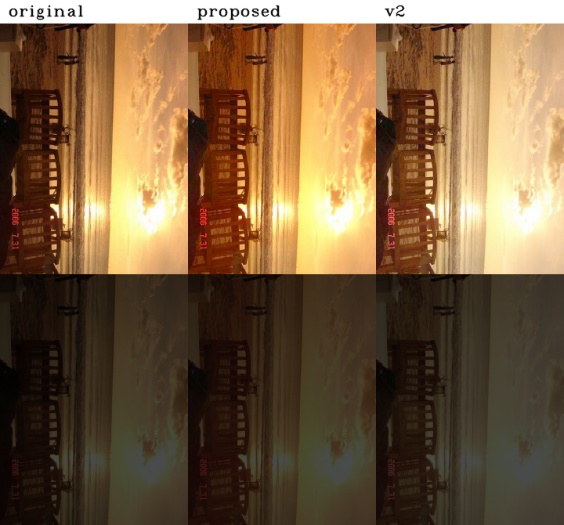
另外，相同的RGB值會轉換成相同的enhanced RGB值，因此可以利用dynamic programming的技巧來減少computation cost，將計算過的RGB值對應的enhanced RGB值存起來。又或者可以存全部的可能性，讓整個過程便成table look up的概念。

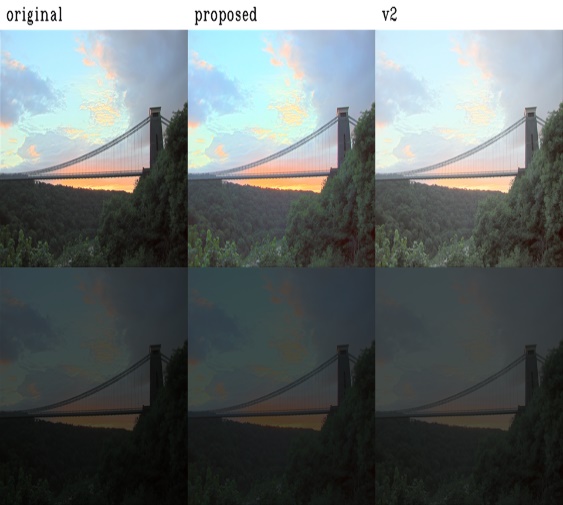
* **Gamma Correction – CIELUV & HSV & YCrCb & RGB**
* **Model Analysis - CIELAB & HSV**
* **HSV Enhancement**
* **Low Light Enhancement**
* **Color Correction**

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。

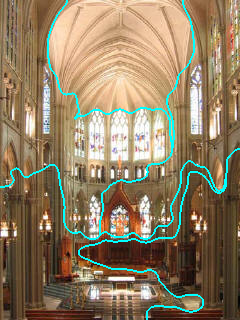
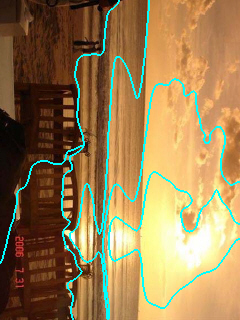
 

從圖片可以看出，整體是有enhancement的效果。但是因為目前參數a是訂值，因此如果a太小的話，暗處的細節就無法獲得足夠的enhancement；但是如果參數a太大的話，會使得較亮or白色的部分變得分不清差別。

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**

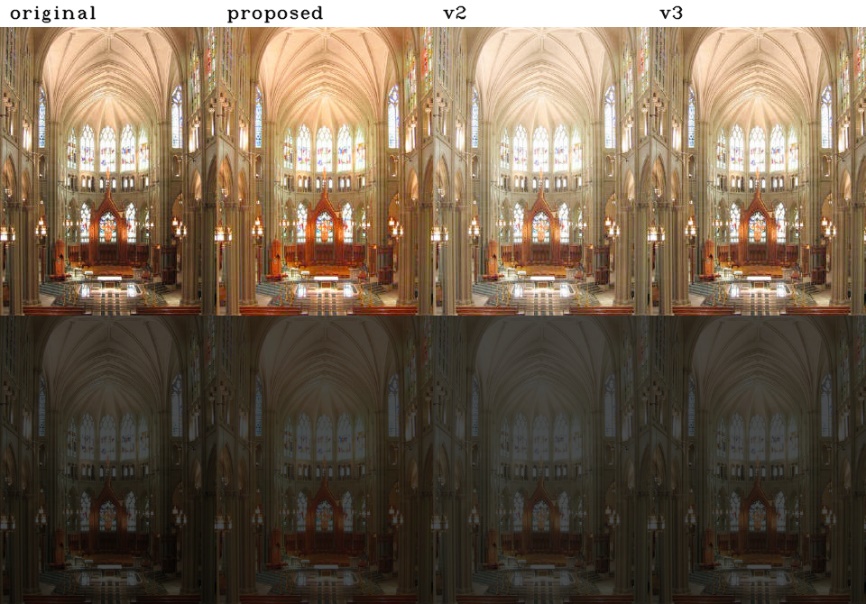
由於前面的方法感覺是可行的，因此希望從此著手，利用super-pixel的概念，將image切割成幾個groups，因為我們認為亮的區域會成一區，暗的區域會成一區，因此不需要分成太多區域，且同一區域可以用相同的參數去做調整。

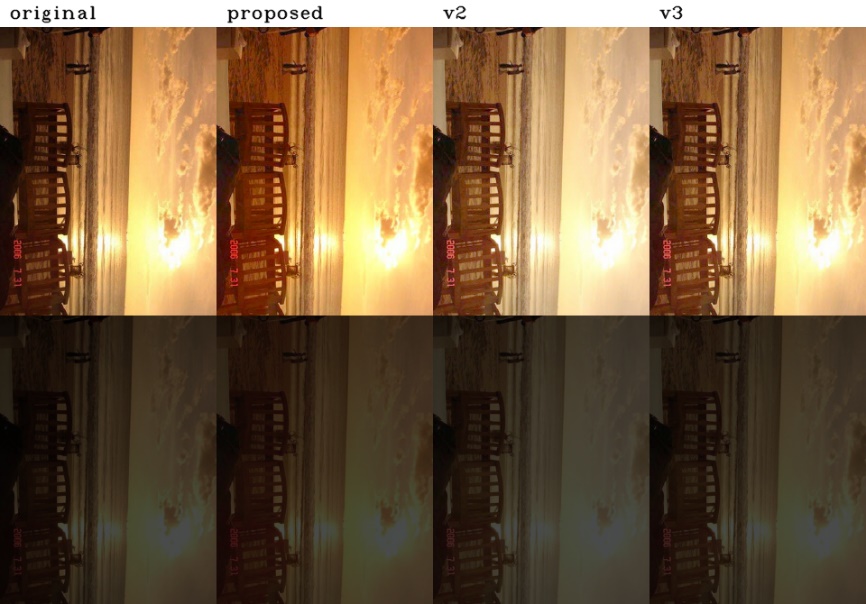
 

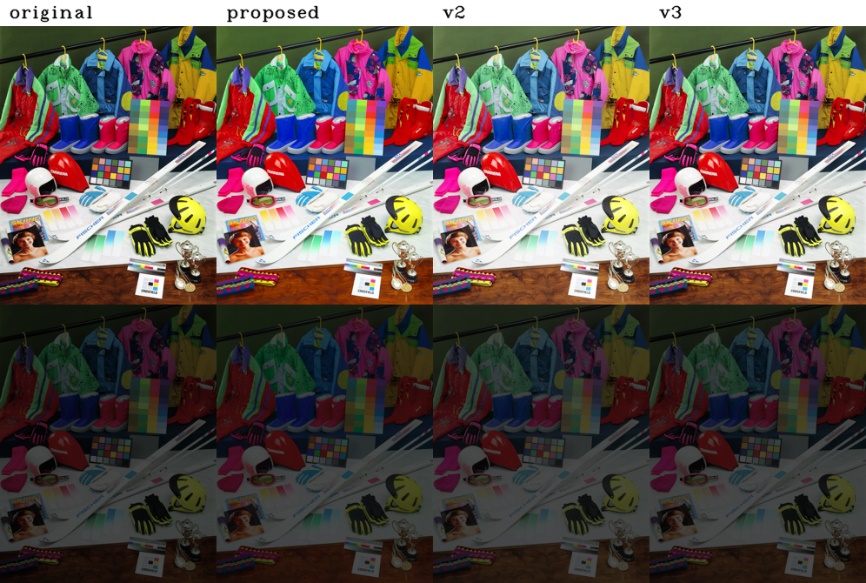
 

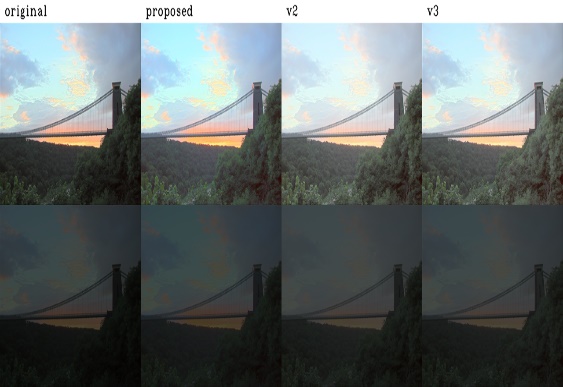
另外，由於我們覺得可以代表一個區域的亮暗程度最方法，還是利用CIECAM02 model來計算lightness & chroma最方便，因此我們先計算super-pixel內的mean color value，接著計算這個mean color value的lightness J & chroma C。最後，由於我們希望亮的部份的參數a值要小，而暗的部分參數a值要大，因此設a = sqrt(1 / (J\*C))，這樣就能夠符合我們對於a的要求。

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。









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。