

Digital Image Processing

Final Project

R06944007 楊喬諳 R06922046 胡柯民 R06944042 蔡健華

題目

天空美化

實作細節

References

- (1) Image部分 : *Color Harmonization*, Daniel Cohen-Or, 2006
- (2) Video部分 : *Color Harmonization for Videos*, Nikhil Sawant, 2008

Source Code

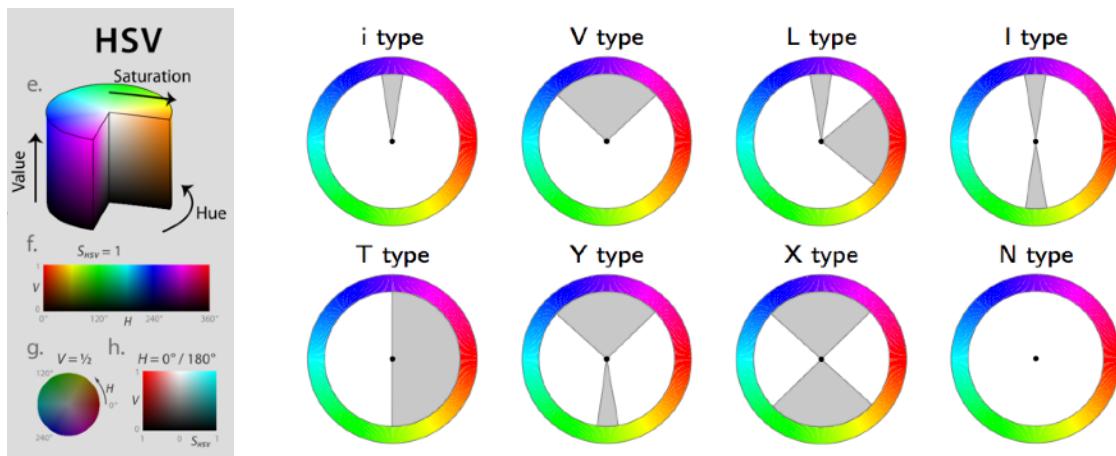
Github: <https://github.com/tartarskunk/ColorHarmonization>

Introduction

輸入 : (1) source video, (2) target image作為style

輸出 : 套用target image的harmonic scheme輸出為output video

在HSV空間中，每一張相片可被歸類到以下任一種harmonic scheme：



一個 harmonic scheme 可被其type m和orientation a來定義。

Approach

(1) 找到一張照片的harmony score $F(\cdot)$

$$F(X, (m, \alpha)) = \sum_{p \in X} \left\| H(p) - E_{T_m(\alpha)}(p) \right\| \cdot S(p)$$



上兩圖為用histogram表示Hue Distance。

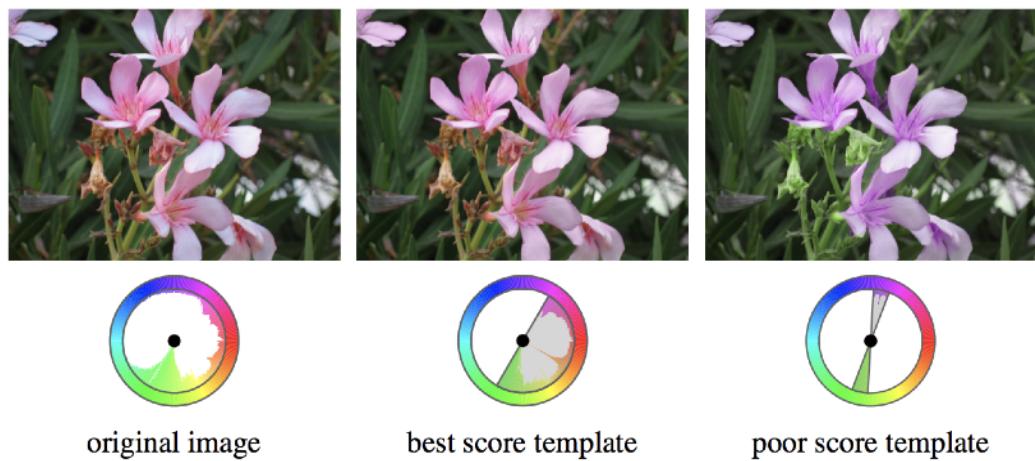
紅圈處為template兩個端點，並將兩端點的範圍內設成0。
以Saturation作為權重。

(2) 找出在單個scheme m 上最佳的orientation α 去minimize $F(\cdot)$

$$M(X, T_m) = (m, \alpha_0) \text{ s.t. } \alpha_0 = \operatorname{argmin}_{\alpha} F(X, (m, \alpha))$$

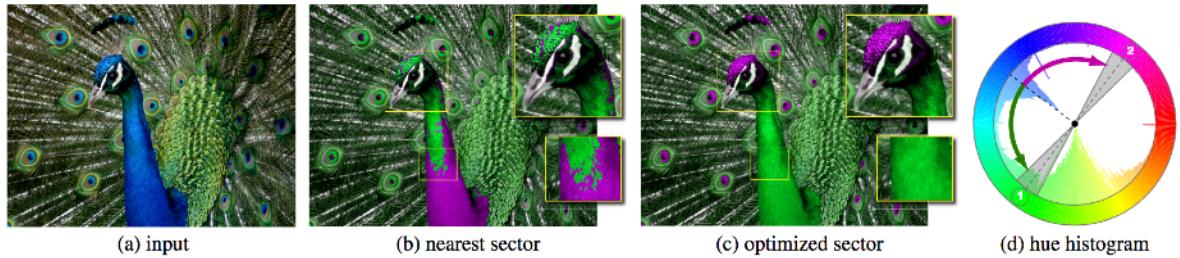
(3) 找出各scheme m 中 最能minimize $F(\cdot)$

$$B(X) = (m_0, \alpha_0) \text{ s.t. } m_0 = \operatorname{argmin}_m F(X, M(X, T_m))$$



上圖顯示套用好壞harmonic scheme所造成的效果。可看出分數高的scheme表現的圖片較符合人所喜歡的色調，而分數低的則是較為詭異的色調。

(4) 處理破圖問題：因為前述方法未考慮spatial information而產生



- a. 把空間上附近的pixels合理分類到同一個sector使影像顏色不會破圖(不連續)
- b. 論文將其歸類為binary classification並設計了energy function以最小化之，實作上發現這樣的方法相當耗時 $O(2^n)$ ，無法快速處理影片，因此我們提出c.的方法。

$$E(V) = \lambda E_1(V) + E_2(V),$$

$$E_1(V) = \sum_{i=1}^{|\Omega|} \|H(p_i) - H(v(p_i))\| \cdot S(p_i),$$

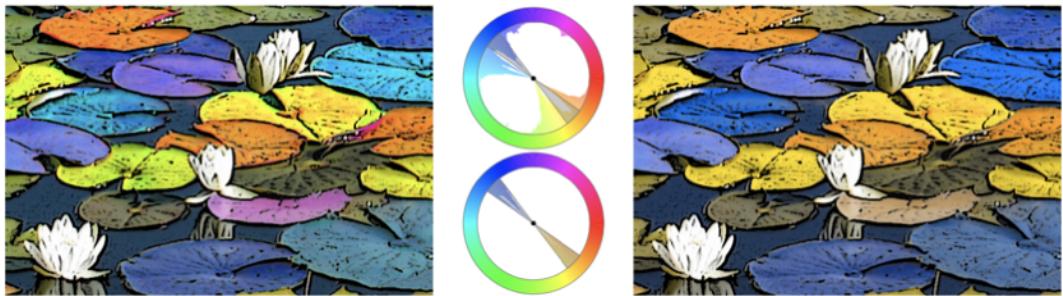
$$E_2(V) = \sum_{\{p,q\} \in N} \delta(v(p), v(q)) \cdot S_{\max}(p, q) \cdot \|H(p) - H(q)\|^{-1}.$$

- c. 使用superpixels技術先去分群，每群內部進行voting，避免同群內的不連續。

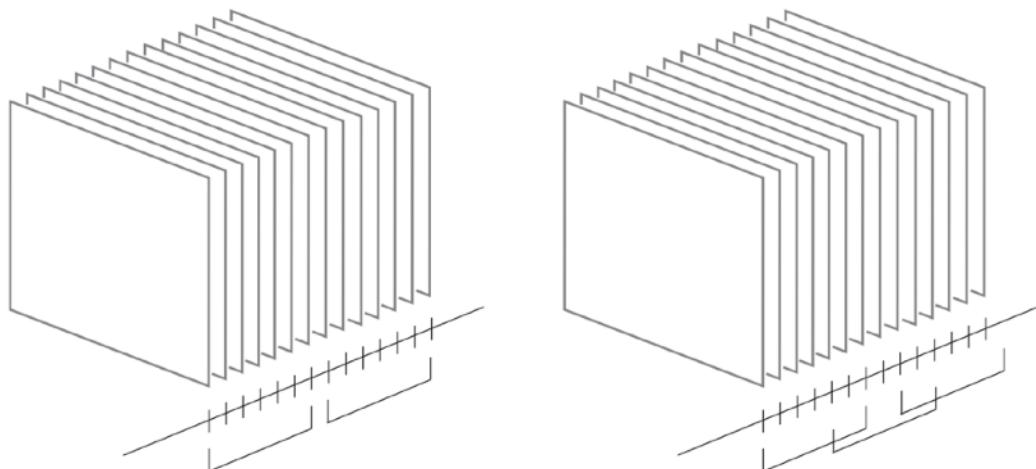


(5) shifting hue

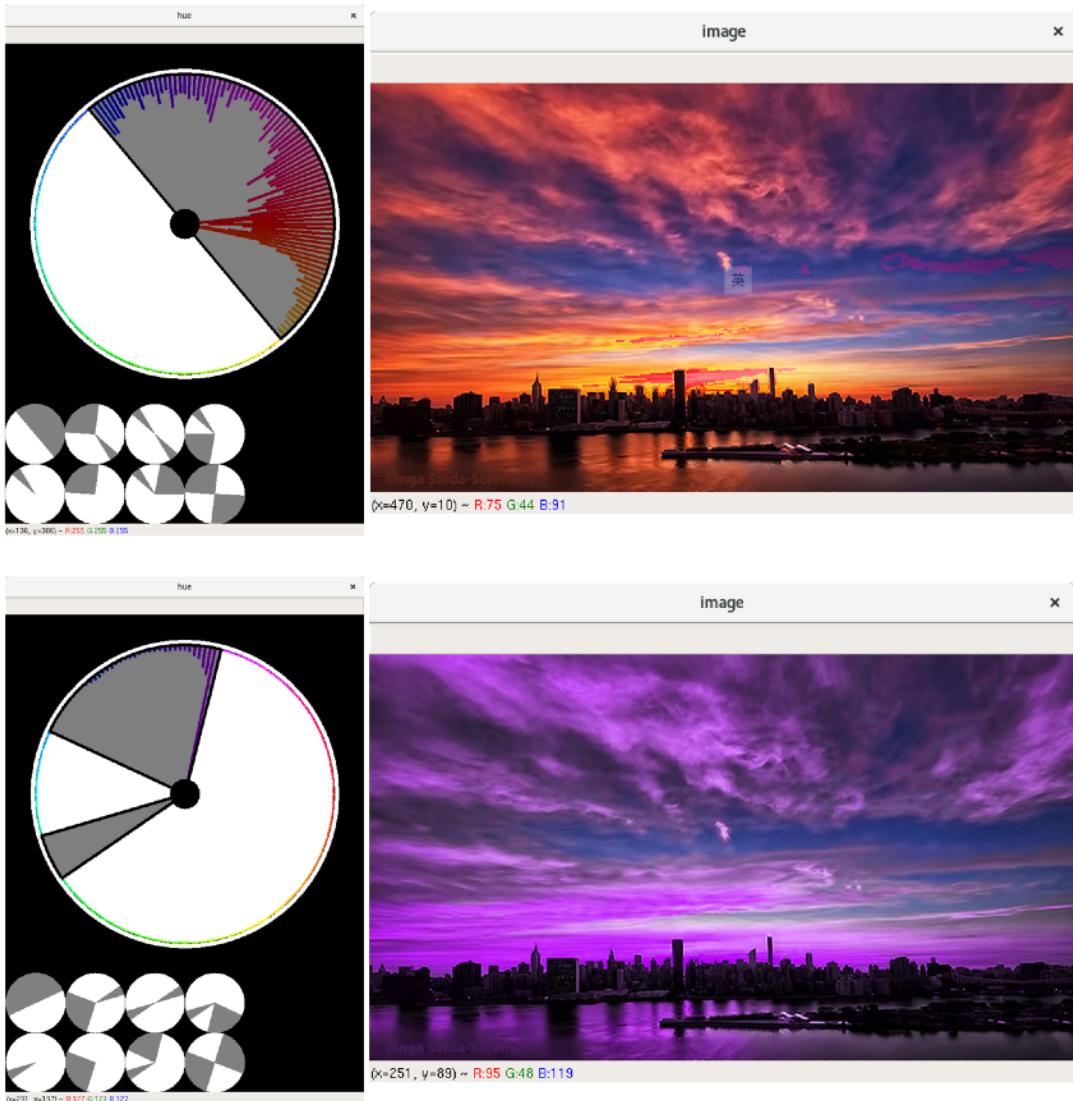
$$H'(p) = C(p) + \frac{w}{2} (1 - G_{\sigma} (\|H(p) - C(p)\|))$$



(6) 把影片的連續frame分為一個group，然後平均他們的hue histogram後算出的 harmonic orientation α ，套用到各自影像中得出個別的harmonic type m 。注意 group 跟 group 之間會有 overlapping，可以減少因 boundaries disjoint 所產生的 flickering。



(7) 實作一個可互動的UI。可以讓使用者看到input image的hue histogram以及套用best harmonic scheme的hue histogram，同時可以操作介面去更動不同的scheme type和orientation。UI截圖如下：

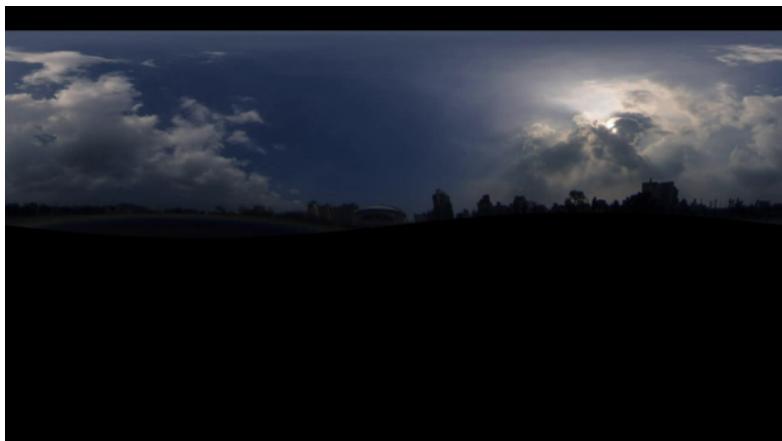


(8) 總結：我們以 Python 實作這篇論文，並設計出可以平行處理(numpy的矩陣運算)的演算法，即使在 Python 這類直譯語言先天上執行較慢的缺點下，也能夠快速計算(一張frame半秒內可轉換harmonic scheme)。並設計了 UI 讓使用者可以互動，得到使用者定義的結果。我們在處理影片上也有不錯的效果。

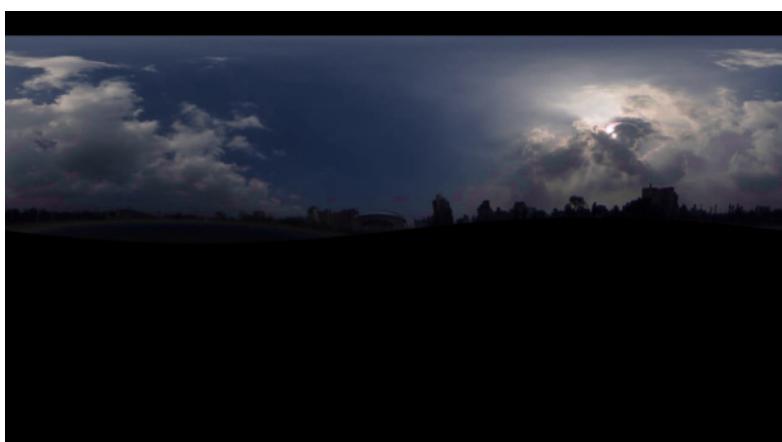
結果截圖

Source: sky_1.mp4

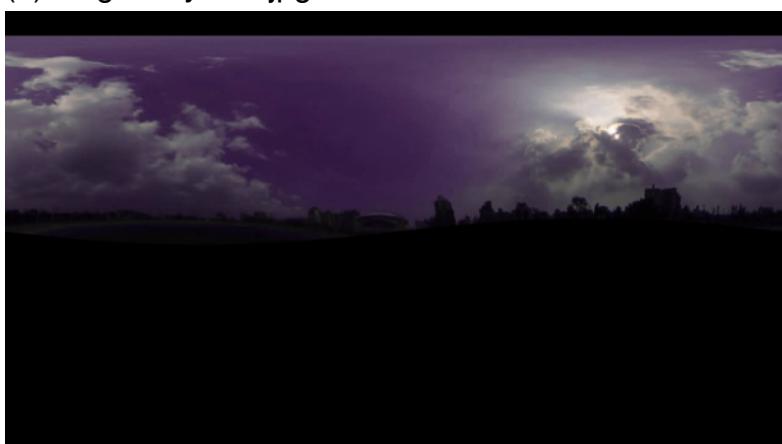
(1) Target: style_1.jpg



(2) Target: style_2.jpg

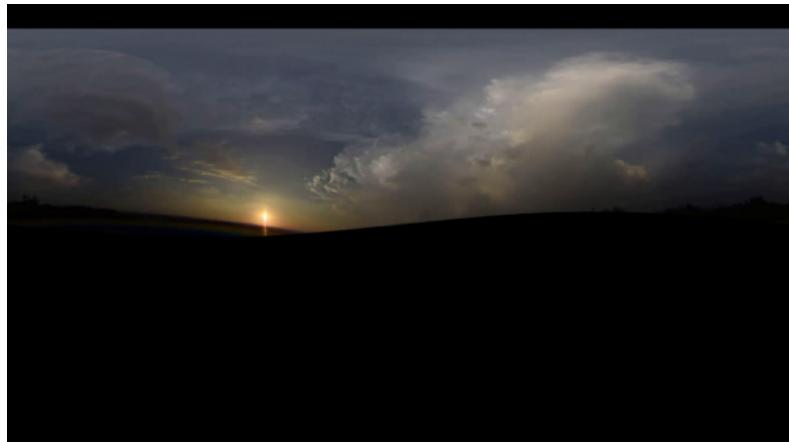


(3) Target: style_3.jpg

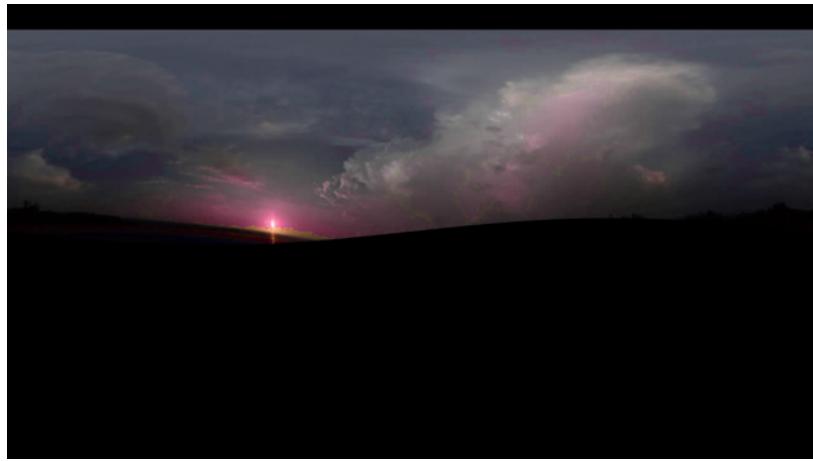


Source: sky_2.mp4

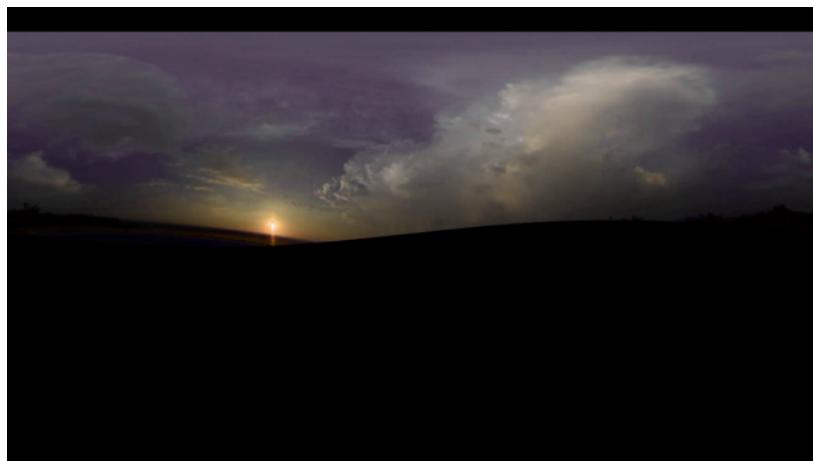
(1) Target: style_1.jpg



(2) Target: style_2.jpg

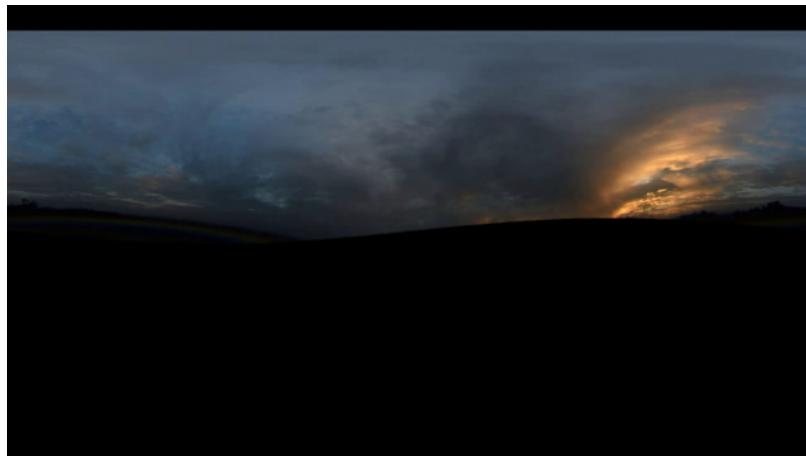


(3) Target: style_3.jpg

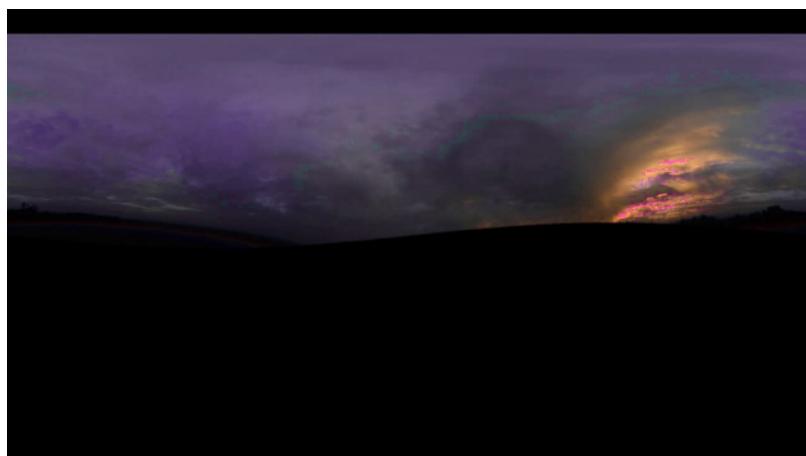


Source: sky_3.mp4

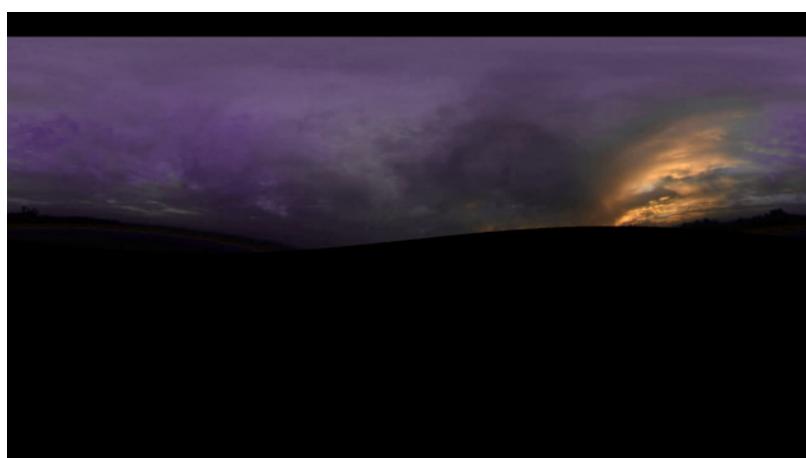
(1) Target: style_1.jpg



(2) Target: style_2.jpg



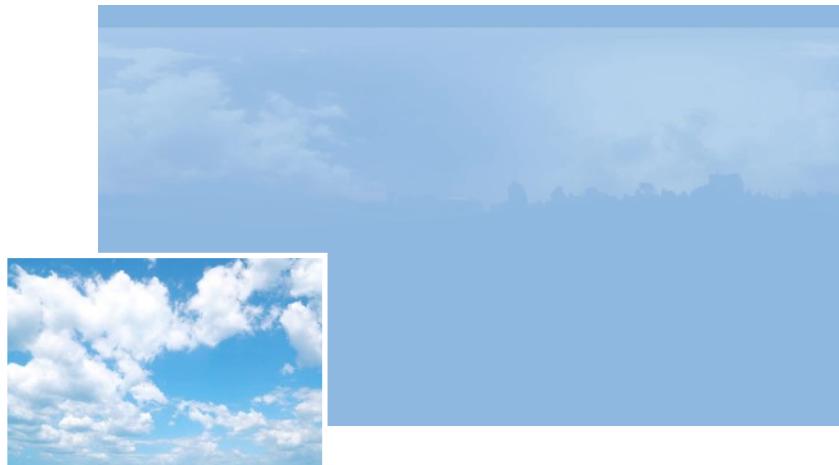
(3) Target: style_3.jpg



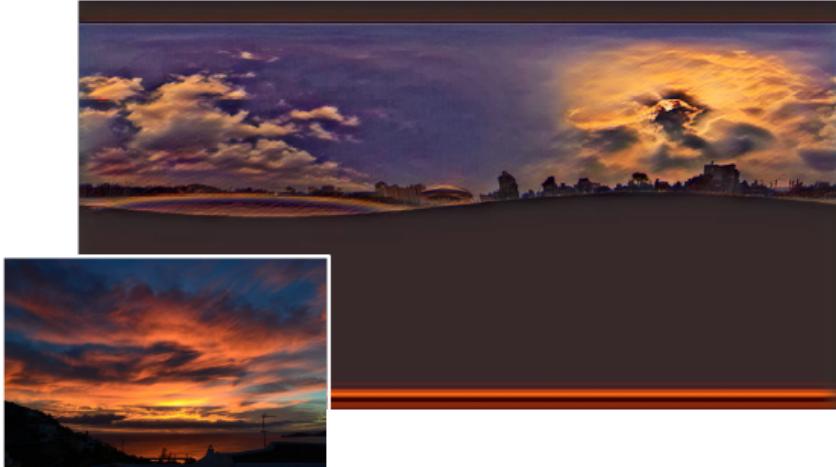
討論

我們另外嘗試了助教提供的2種方法。

(1) Color Transfer。我們發現做出來的效果並不是很好，會將黑的地方也改動成style的color。如下圖：



(2) Artistic style transfer for video。若直接拿來套在影片上時，由於缺少考慮frame及frame之間的consistency，容易出現閃爍的現象。



(3) Color Harmonization。這是我們最後選用的方式。由於只處理frame本身的hsv值，並不會影響原本的構圖，所以閃爍的現象減少許多。我們以 Python 實作該篇論文，並設計出可以平行處理(numpy的矩陣運算)的演算法，即使在 Python 這類直譯語言先天上執行較慢的缺點下，也能夠快速計算(一張frame 半秒內可轉換harmonic scheme)。並設計了 UI 讓使用者可以互動，得到使用者定義的結果。