

Description:

整個程式主要分為以下步驟：

1. Load images and exposures:

將拍好的照片，利用os與cv2兩個套件取出，並存進一個list之中，形成一個image list來提供後續使用。同時將image list中每張照片對應的exposure存入另一個exposure list中。

2. MTB alignment:

接下來就可以使用threshold來創建binary image。對於每對影像，先將它們的灰度影像進行二值化處理，將得到的binary image存在 binary_images這個list中。然後利用cv2.matchTemplate在每對影像之間的binary image進行template matching，以計算出它們之間的偏移量。最後根據每個影像與其它所有影像之間的偏移量計算出總偏移量，以實現對齊操作，得到對齊後的影像。

3. Sample pixels

得到對齊的圖像後就抽樣指定數量的pixel來進行運算。挑選的方式是：

首先將每個區域設為 25×25 ，計算每個區域內像素值的標準差，然後選擇標準差最大的像素作為代表。每次都從一個區域選出像素代表，直到選出的樣本數到達指定數量。這些選中的像素組成了一個list。接著函式會對每個選中的像素，從每張圖片中取出該像素的值，存在一個叫samples的list中。

最後將samples return。

4. Construct HDR

Part1:

建立一個新的函式solve_respond_curve，input為 (Z, B, l, w) Z為不同照片的每個選點的強度，B為log exposure time，l為lambda，w為weight function。

函式中宣告新的a,b矩陣，初始值都為0，由於此步驟的目的為得到inverse response function，我們需要線性系統的解中的前256項，也就是g。其中填矩陣內容的部分依照上課講義中的matlab code來修改為python版本。

Part2:

在得到g後，我們就可以用它來找出場景中所有pixel的真實強度值，再將這些值存進E這個陣列。我們建立函式radianc()，利用part1中得到的g以及定義好的weight function跟原本有的exposure time，帶入課程講義中第46頁的公式，得到最後的結果。

Part3:

這一步的目的是要得到false-color radianc mapping的圖。由於不同顏色channel會產生不同的g function，因此我們設計參數中帶入三個不同的g。show_false_color函式中的hdr為一個3維陣列，儲存了不同color channel中的所有pixel的真實強度值。又因為radianc()回傳的陣列為一維的，所以再存入hdr時需要reshape回原來照片的長寬。

在視覺化radianc mapping的部分，我們先將hdr中所有pixel的radianc值正規化，避免無法讓所有radianc出現在false-color mapping中的狀況。再

將hdr中的radiance data轉為gray scale來檢視融合3個channel後每個pixel的相對強度，最後選擇用openCV提供的colormap ‘jet’來顯示出radiance的分佈圖，radiance數值小到大以顏色深藍到深紅來顯示。



5. Tone mapping

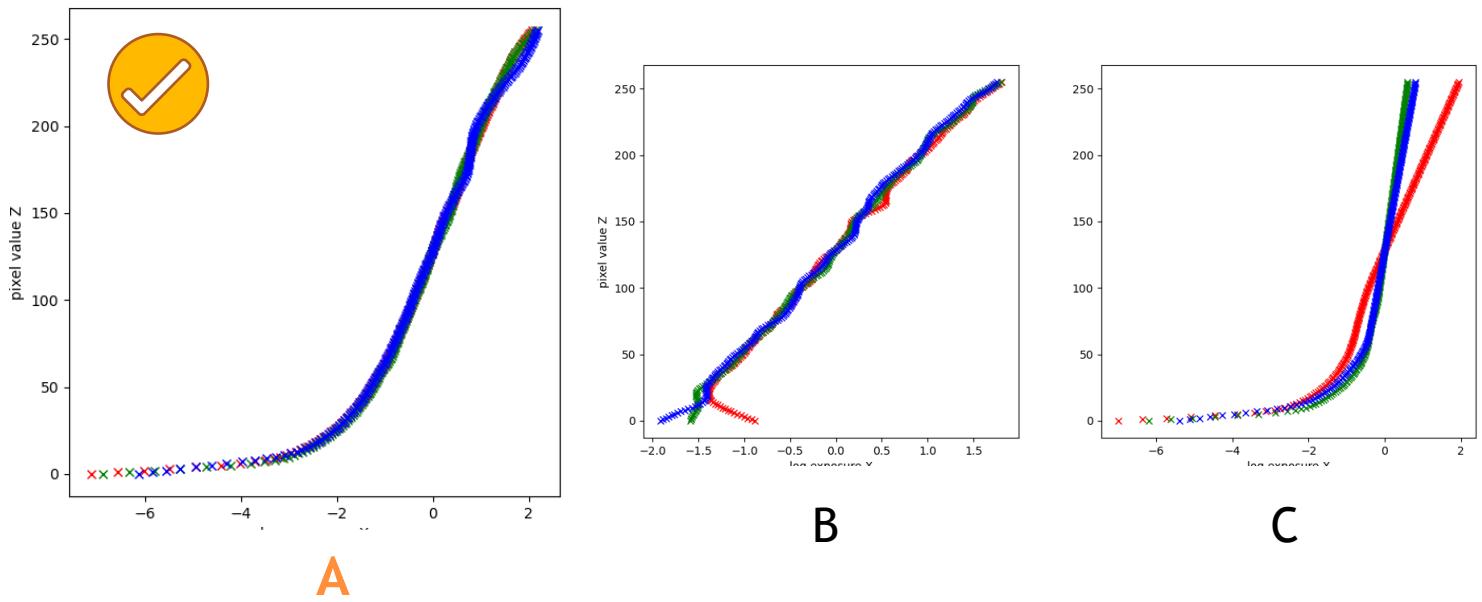
利用openCV的tone mapping套件，將HDR image的pixel value縮至0到255之間，將HDR image轉換為8位元的LDR圖像，輸出最終的result.png。

Experiment and Comparison

1. response curve

我們總共試了三種選點方法，(A)第一是隨機選點，也是最簡單直覺的方式，(B)第二是以每個pixel在不同曝光時間的照片中亮度(0-255)的標準差大到小來排序，選出前500個pixels，(C)第三種是將每張圖片分成 25×25 的好幾塊區域，計算每個區域內像素值的標準差以來排序，以來確保每個區域中都可以選到一些pixels。

比對了三種選點方式後我們決定選用第一種，因為用它做出來的hdr成果最佳。我們用不同選點方式，選出500個pixels算出的response curve如下：

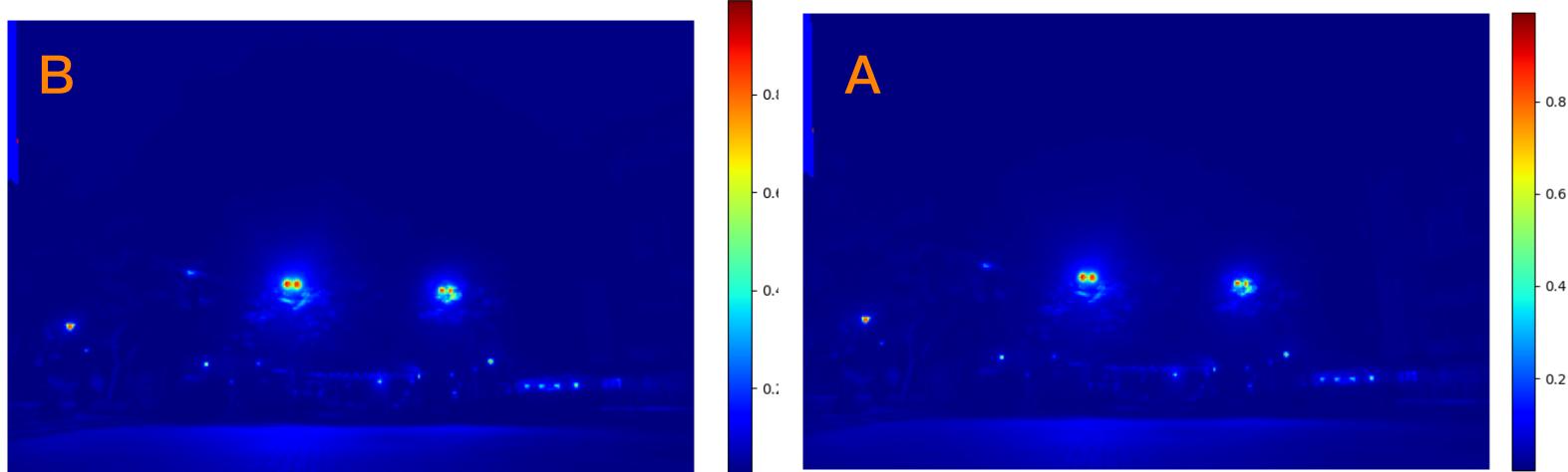


2.Radiance map

另外，最後在使用tone mapping函式的時候，因為我們是使用reinhard的演算法，可以調整四種參數，包括影響了照片亮度、色彩及contrast等等。這些參數都會影響最後的照片效果，因此調配最適合這個場景的參數也是一門需要不斷測試的過程。

Radiance map在做出hdr image的過程中是很重要的一步驟，因為在這步之前我們全部都是以程式碼去測試自己的演算法有沒有成功，但以color map來呈現的話可以清楚知道演算法之間的差異與好壞。

我們用三種選點方式去實作後，發現(A)與(B)選點方式的結果沒有差很多，但是唯獨(C)的radiance map看得到更多細節，應該是因為這個演算法強調了一些光度差異較大的點，因此算出來的radiance map的亮度會普遍高一點。



3. Tone mapping

我們調整出tone mapping函式中的最佳參數，以將場景中的最完美的呈現出來。在過程中我們發現，如果是用前面提到過的(B)選點方式，那呈現出來的視覺效果會非常的過曝，我們推測是因為(B)這個演算法單純只是將標準差最大的前幾個拿出來處理，但這樣的話可能會讓選到的點集中在特定區域，所以算出來的curve以及最後的hdr照片都不太理想。(A)的色彩正確度及曝光程度則是視覺上看起來最佳的。

隨機選點(A)效果比較好的原因可能是：

標準差最大的像素點不一定是最能代表區塊整體特徵的像素點。選擇標準差最大的像素點可能會導致HDR合成過程中出現較大的噪點或色差，從而影響最終合成結果的質量。

相反，隨機選擇像素點可能會更好地代表區塊的整體特徵。隨機選擇可以幫助我們避免選擇那些在某些特定情況下不太適合的像素點，從而提高HDR合成的質量。

另外，最後在使用tone mapping函式的時候，因為我們是使用reinhard的演算法，可以調整四種參數，包括影響了照片亮度、色彩及contrast等等。這些參數都會影響最後的照片效果，需要不斷測試哪種組合最適合這個場景。





A (the best)



B



C