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。

1. MTB alignment:

接著利用MTB algorithm將image list中的圖片對齊，使它們在相同位置上有相同的像素值。首先計算每個圖片的灰度版本，將其存儲在 gray\_images這個list中。接著計算圖片灰度值的中位數，作為計算的閾值，並根據每個像素是否大於閾值，將每個像素標記為 0 或 1。接下來使用binary bitmap來計算位移矩陣。對於每個圖像和每個位元，計算當前位圖和前一位圖之間的差異，並使用差異值計算位移向量，以獲得最終位移量。最後使用得到的位移矩陣將原始圖像對齊，獲得對齊後的圖像列表。

1. 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100x100，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。

1. Construct HDR

*P****art1:***

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。

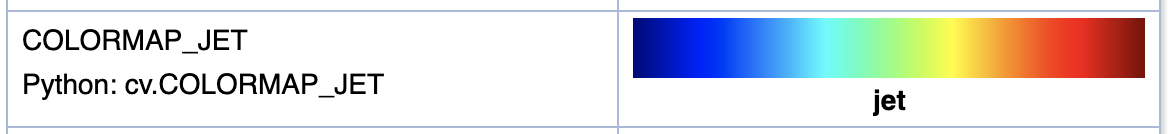
函式中宣告新的a,b矩陣，初始值都為0，由於此步驟的目的為得到inverse response function，我們需要線性系統的解中的前256項，也就是g。其中填矩陣內容的部分依照上課講義中的matlab code來修改為python版本。

***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*radiance()*，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 radiance 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radiance()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radiance mapping的部分，我們先將hdr中所有pixel的radiance值正規化，避免無法讓所有radiance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。

1. 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

我們最後算出的response curve如下:

Radiance map如下:

Tone mapping後的結果如下: