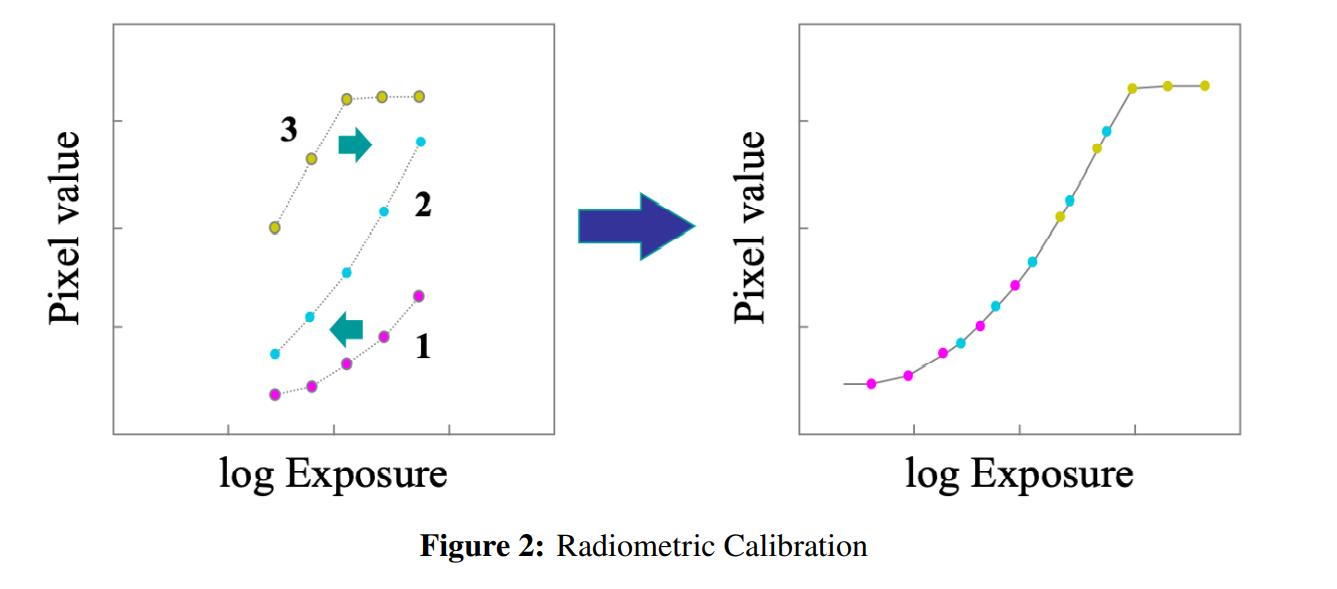
* Experiments a.
  + What is the meaning of this figure?

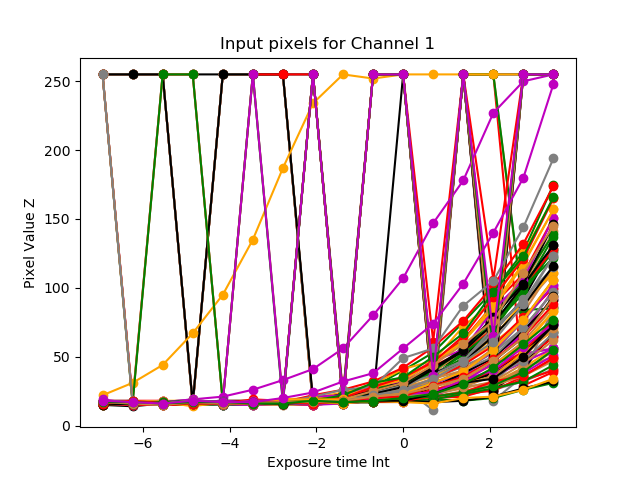
Figure 2之左圖為多張不同曝光時間的bracketing images的pixel值與曝光時間取log的關係。同一條線代表各張不同曝光時間的bracketing images之同一個pixel位置的連線，以左圖為例，因為每條連線有五個點，所以有五張bracketing images，而三條連線表示有三個不同位置的採樣pixel。可發現曝光時間越長，pixel value值越大，呈現正相關。

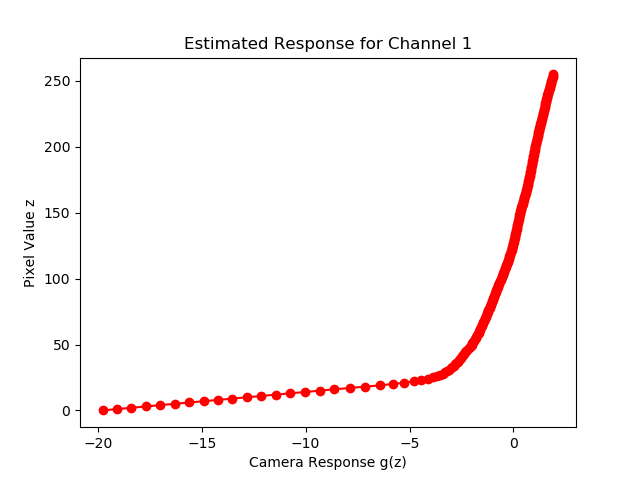
欲求得Figure 2之右圖，需先將Implementation中1.1 Camera response calibration中的equations用numpy.linalg.lstsq()求出後，再以(本來就是已知的pixel value)與(Camera response)作圖即可得到Figure 2之右圖。從Figure 2之右圖可發現，當曝光時間一直變大，並且到達一個閾值之後，其pixel value都會落在256並且不再增加。

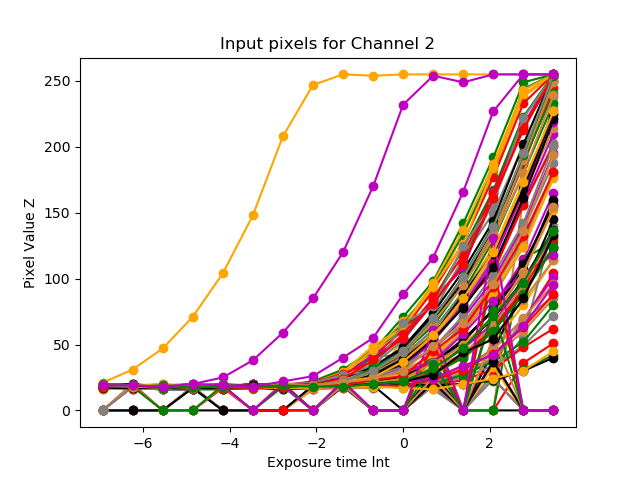


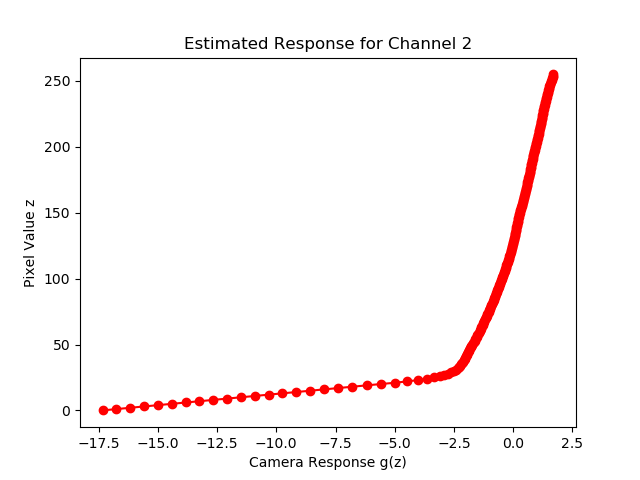
* + Try to plot a same figure for memorial image set

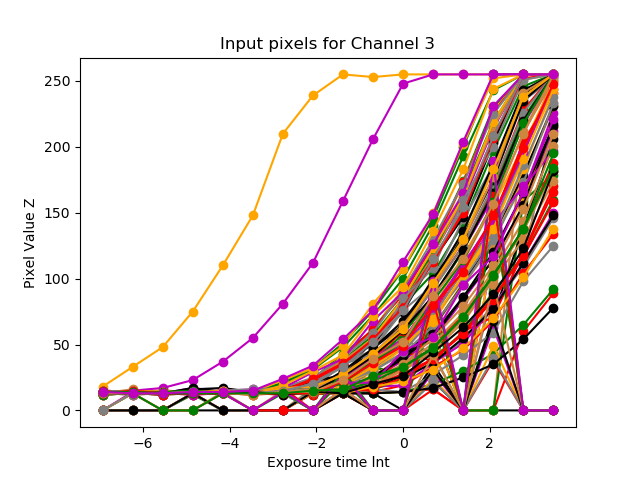
下面六張圖片儲存在”Experiments/a/”資料夾中，並會在執行cr\_calibration.py時自動秀出並儲存。

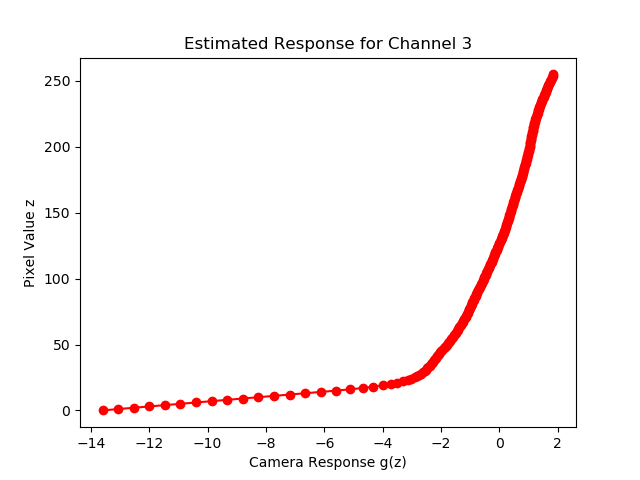








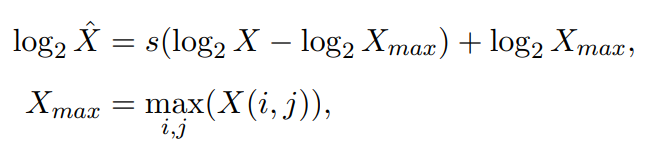




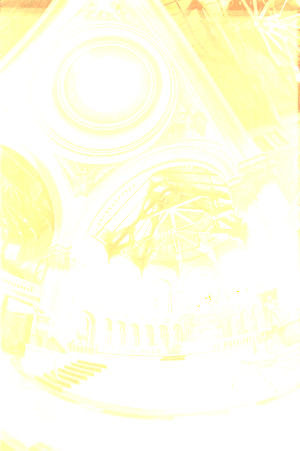
* + How do you plot the figure and why do you choose to do so?

首先將16張不同曝光時間的照片之pixel value讀取出來，16張照片分別各有768\*512個pixel，並且都是3 channels，接著對這些pixels位置做採樣，採樣方式為二維每隔64個pixel位置採樣一次，故這樣總共會採樣出(768\*512)/(64\*64) = 96個pixel的位置，換句話說這16張不同曝光值的照片之3個channels的採樣點都是這96個pixel位置。接著用各個channel採樣出來的pixel values(16張照片\*96個pixel位置)，讓16張相同pixel位置的pixel value(16,)對其曝光時間(16種曝光時間)作圖得一連線(故1條連線上有16個點)，並且用迴圈作96次，可得96個連線，即可得到”Input pixels for Channel X”的圖。然後將function estimateResponse()回傳的Camera Response g(z)對Pixel Value z(0-255)作圖得”Estimated Response for Channel X”。另一種方式得到”Estimated Response for Channel X”，是透過function constructRadiance()得到再對取對數得ln，並採用(ln+ln)對Pixel Value z(0-255)作圖亦是一種方式，但是我設計的function constructRadiance()時間複雜度較高，所以我沒有採用此做法。

* Experiments b.
  + Scaling factor in global tone mapping



Implementation中1.2(Global tone mapping)設定的s值為1，也就是最後得到的等於*X*，而當s值為 0到1之間時，log(X)-log()<log()- log()，所以大於*X*，也就是得到的圖亮度較原本高，且s越接近0得到的圖越亮。另外當s值大於1時，得到的圖就會越暗，且s越大會越暗。下圖分別為s=0.6, 0.8, 1.0, 1.2, 1.4的實驗結果，實驗程式碼在tm.py，實驗圖片在”Experiments/b/”資料夾中。

s=0.6

s=0.8

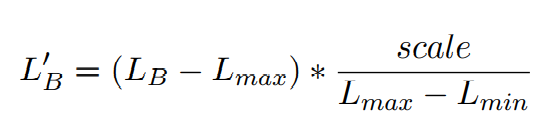
s=1.0

s=1.0

s=1.2



* + Scaling factor in local tone mapping with Gaussian filter



從上式可以發現，為大於等於零的固定值， 所以在scale大於等於1的情況下，是一個小於等於零的值，故scale越大，的絕對值就越大，也就是就越小，並且在detail layer不改動的情況下，Reconstruct color map C就會越小，再套一層gamma correction也會越小，故亮度較暗，；反之若scale越小，則亮度會變亮。

而且可以發現是各個pixel位置都做一次上式的運算，故比較小的pixel位置，較大，此時的絕對值就越大，也就是越小，換句話說就是同一張圖越暗的區域，經過local tone mapping變暗的幅度會越大。下圖分別為s=2,6,10的實驗結果，實驗程式碼在tm.py，實驗圖片在”Experiments/b/”資料夾中。

s=2



s=6



s=10



* Experiments c.
* Free study – Problem 3
* Free study – Problem 8
  + Assumption

Global tone mapping是將三個channel的radiance分別在log domain做線性壓縮，當scaling factor的值越大，radiance被壓縮得越多，也就是會變得越暗。而White balance作用是調整每個pixel value之RGB三色在’known to be white region’的比例，讓這樣的色彩比例在任何光源打進來時，看起來還是有白色的感覺。我覺得這兩個function彼此是互不干涉，所以推測Global tone mapping & White balance在實作上誰先誰後的結果都一樣。

* + Justification

為了驗證我的假設，我將原本test\_HW1.py內的test\_globalTMwb() function流程做修改，從原本的White balance -> Global tone mapping -> gamma correction改成Global tone mapping -> whiteBalance -> gamma correction，兩個與結果與golden相比所得之PSNR都是51.08622831892766，可見Global tone mapping & White balance在實作上誰先誰後的結果都一樣。下面兩張圖是我的實作結果。程式碼在/code/free\_study\_p8.py，結果圖在/Free\_Study/problem\_8/資料夾中。

-------------------------------------------------------------------------------------------

White balance -> Global tone mapping -> gamma correction:



-------------------------------------------------------------------------------------------

Global tone mapping -> whiteBalance -> gamma correction:



-------------------------------------------------------------------------------------------

需特別注意的是，gamma correction必須特別拉出來做，不能放在Global tone mapping中，否則順序錯誤無法得到兩張圖一模一樣的結果。