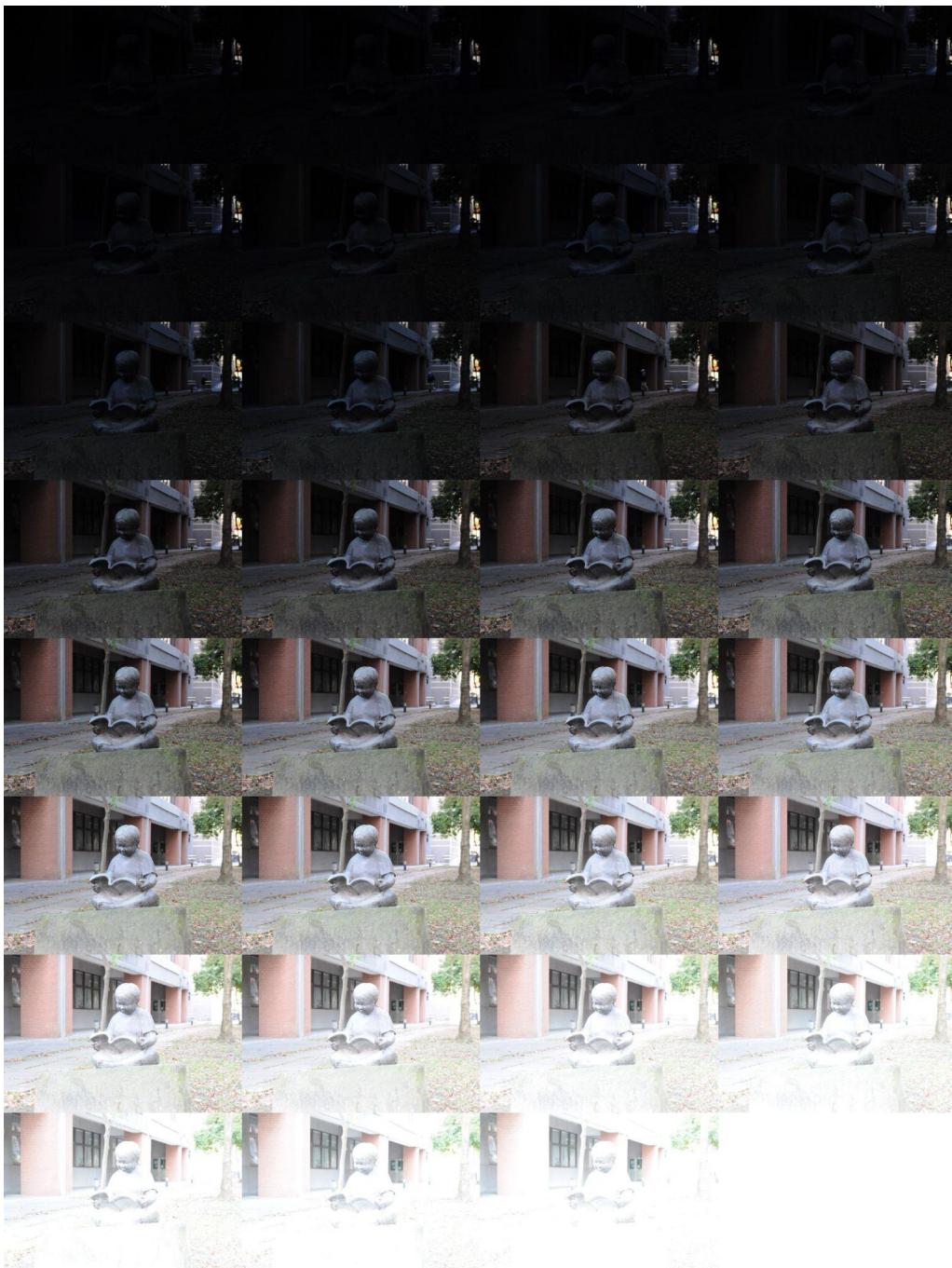


# High Dynamic Range Imaging

Team 35 蔡政諺 郭蕙綺

## 1. Taking Photographs

我們使用助教提供的 Nikon D90, 對同一場景拍攝 31 張不同曝光時間的照片。此外，我們用一個 txt 檔 (data/\*.txt) 記錄每張照片的 shutter speed, 也就是曝光時間的倒數。



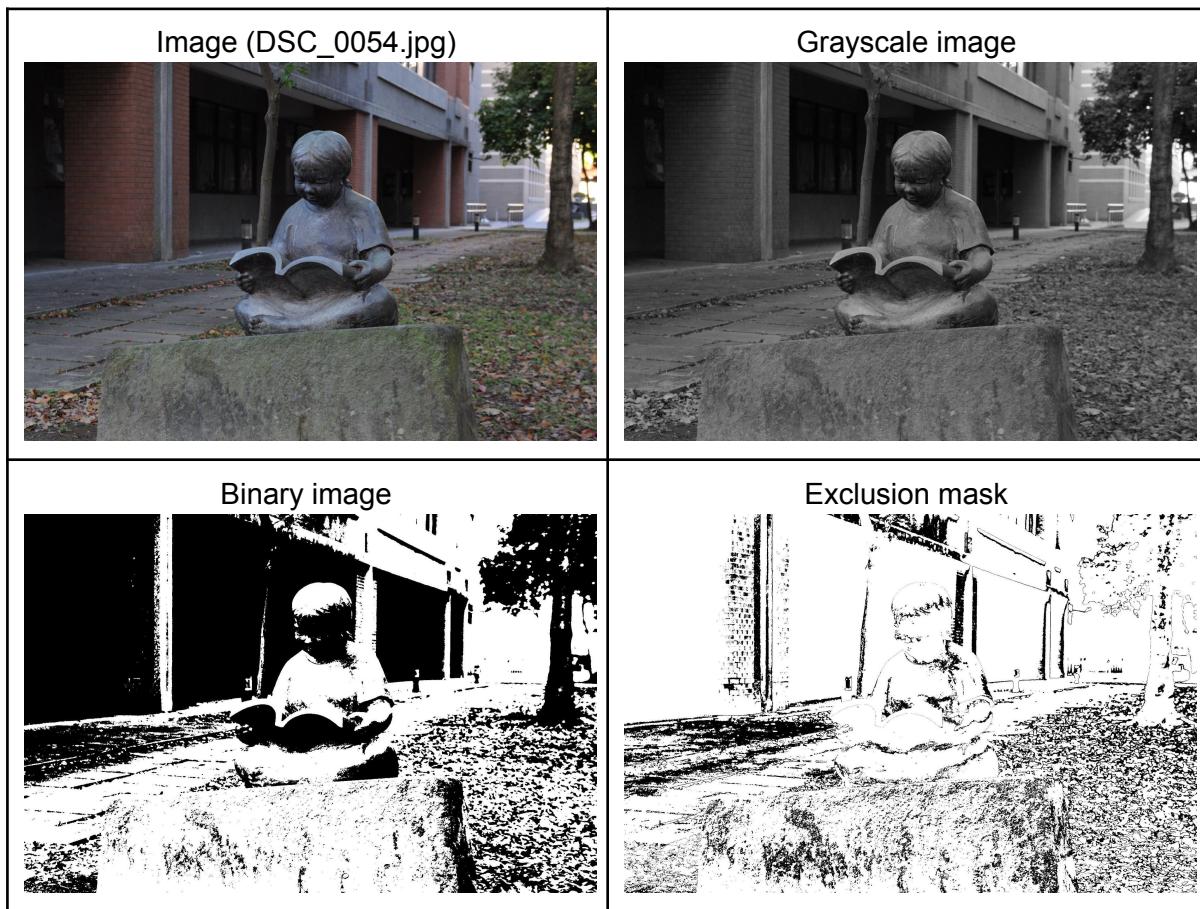
## 2. Image Alignment

由於拍攝時可能因為些微晃動導致照片沒有對齊，我們實作 Median Threshold Bitmap (MTB) Alignment 將照片兩兩對齊。MTB 演算法會對影像做不同比例的縮放（我們的實作中是 1 倍、 $1/2$  倍、 $1/4$  倍、...、 $1/32$  倍），並從最小的縮放比例開始計算 offset，依序往較大的縮放比例將 offset propagate 上去。

在每個縮放比例下，第一步要將 RGB 影像轉成灰階影像，這邊使用上課公式：

$$Y = (54R + 183G + 19B) / 256$$

接著分別計算兩張灰階影像的中位數做為 threshold，將影像轉換成 binary image。由於灰階值太接近中位數的 pixel 可能是 noise，此處會額外輸出一個 exclusion mask，將  $threshold \pm 10$  的像素標記為 0，其餘為 1。



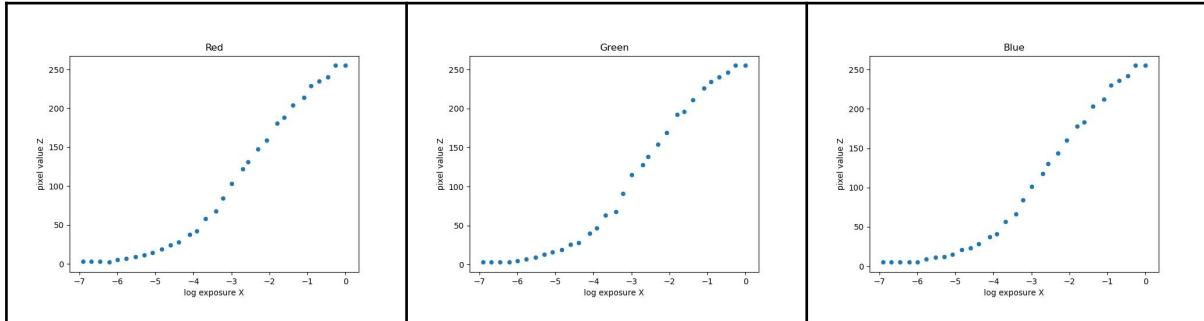
接著計算往鄰近的九宮格做位移後，兩張 binary image 有多少 pixel mismatch，這邊同樣使用上課公式：

$$\text{loss} = (\text{im1} \text{ XOR } \text{im2}) \text{ AND } \text{mask}$$

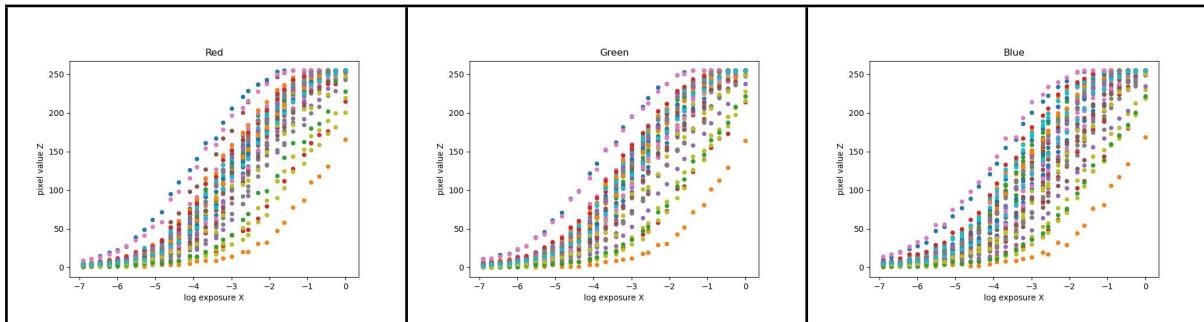
意思是如果兩張 binary image 的數值不同，而且沒有被 exclusion mask 排除，則視為錯誤。九種位移下，造成的 mismatch 最少的 offset 會被選為最終結果，並提供給下一個縮放比例做為九宮格中心。反覆做到 1 倍縮放時，就完成兩張影像的 alignment。

### 3. HDR Reconstruction

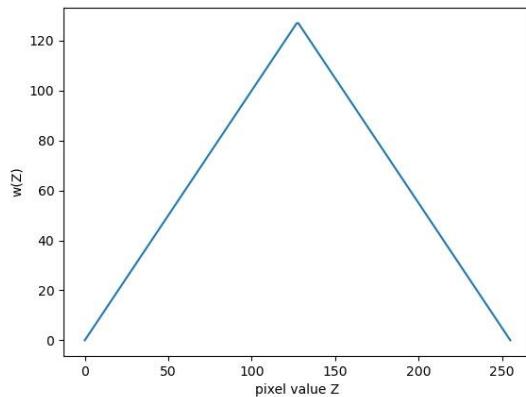
下一步要估算相機的 response curve, 就可以根據像素值回推場景的 irradiance, 我們用上課介紹的Debevec演算法來實現。首先要隨機取樣N個像素位置, 每一個取樣的位置共有31個點, 代表每個曝光時間下的像素值, 下圖是N=1的例子。



根據講義我們設定N=50, 如下圖所示, 每個顏色代表一個取樣位置。Debevec演算法的目的就是水平位移這些不同顏色的曲線, 並找到一個函數g能達到最小平方差。由於這樣會有無限多組解, 所以還要增加g(127)=0的限制。

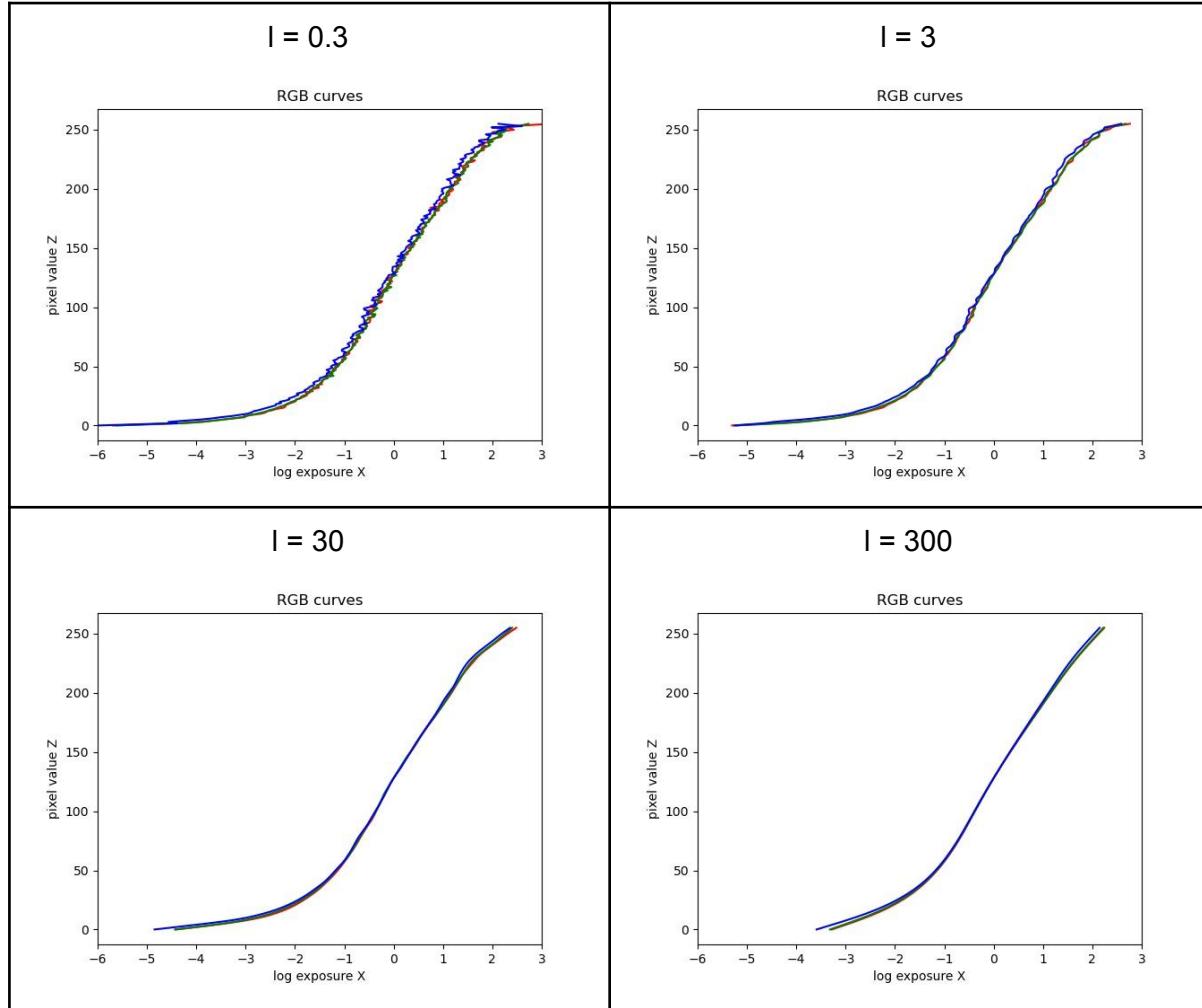


我們採用hat weighting function 當作像素值的 weighting function, 使較極端的數值(例如0、255)的權重變小。



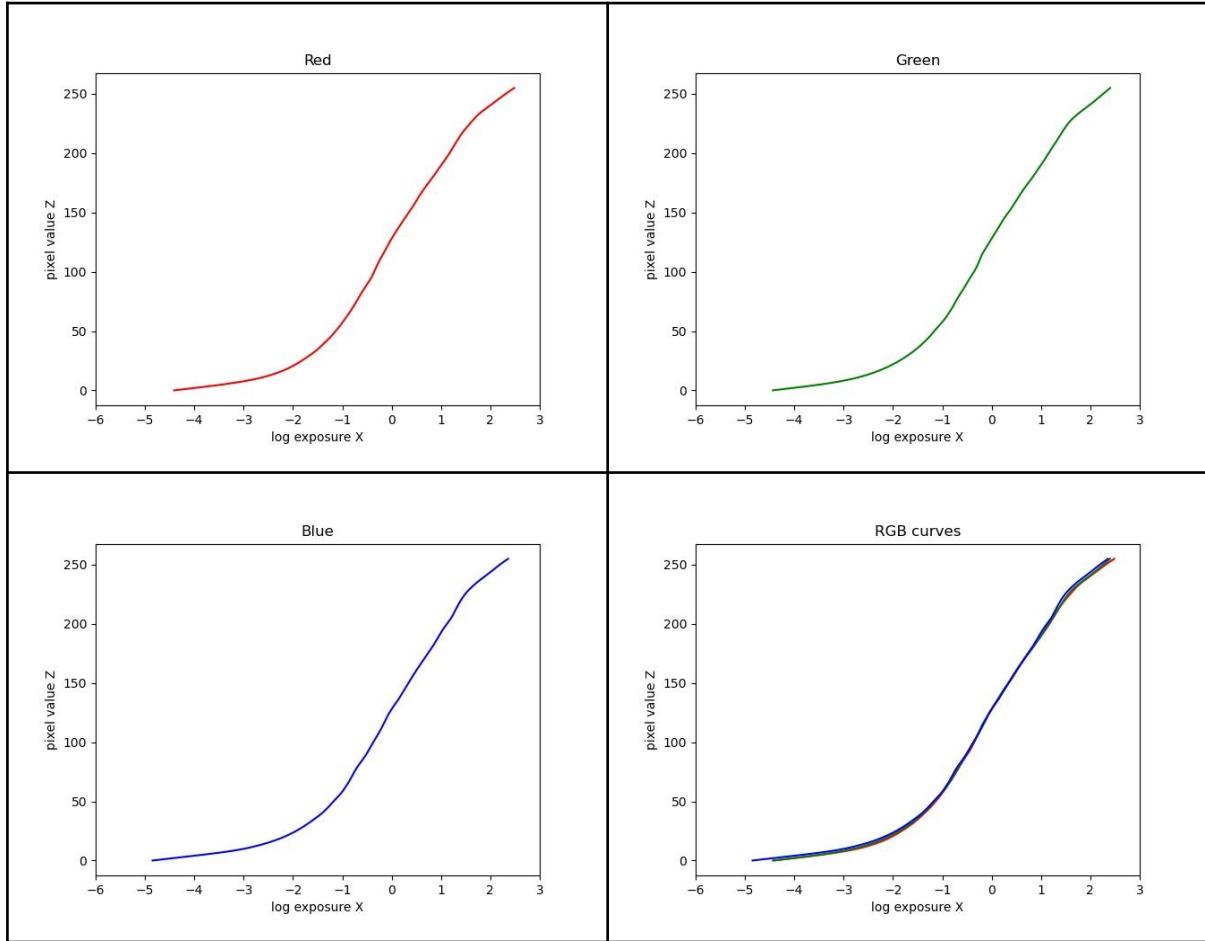
此外, 還要設定參數l, 決定smoothness的程度。l的數值愈大, 曲線就會愈平滑。下圖呈現了l=0.3、3、30、300四組數值得到的response curve, 可以發現l=0.3、3的時候, 曲線會呈現

鋸齒狀，也就是有時候像素亮度變亮時，回推的 irradiance 反而下降，並不合理；但  $I = 300$  的時候，曲線有 oversmoothed 的現象，可能會不符合資料真實分布。因此最後我們設定  $I = 30$ 。



接著將講義的 Matlab code 轉成 Python 語法，建立 linear system ( $Ax = b$ )。由於這是一個 overdetermined system (#row >> #column)，我們使用 numpy 的 SVD 套件求得矩陣  $A$  的 pseudo inverse，進而求得  $x$  的解。對於 RGB 三個 channel，我們都可以分別算出一條 response curve。

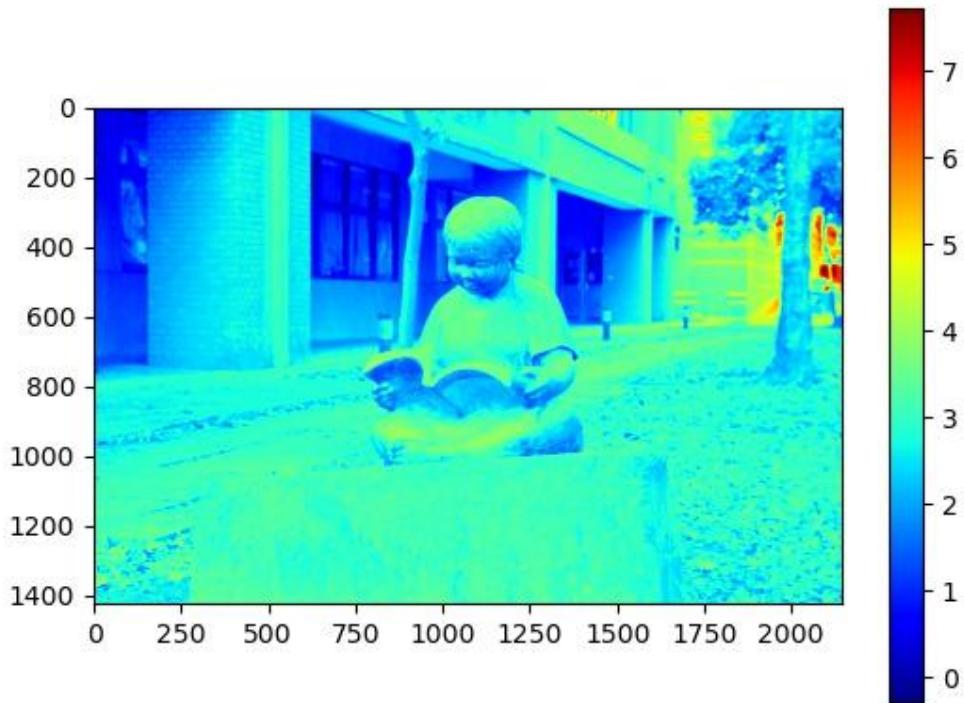
- Response Curves



有了 response curve, 就可以用下列公式, 算出每個像素的 irradiance (log domain), 重建 radiance map:

$$\ln E_i = \frac{\sum_{j=1}^P w(Z_{ij})(g(Z_{ij}) - \ln \Delta t_j)}{\sum_{j=1}^P w(Z_{ij})}$$

- Radiance Map



## 4. Tone Mapping

在 tone mapping 的部分，我們使用了 cv2 內建的 Tonemap、老師提供的tmo package、自己刻的 Bilateral Filter algorithm。

- cv2

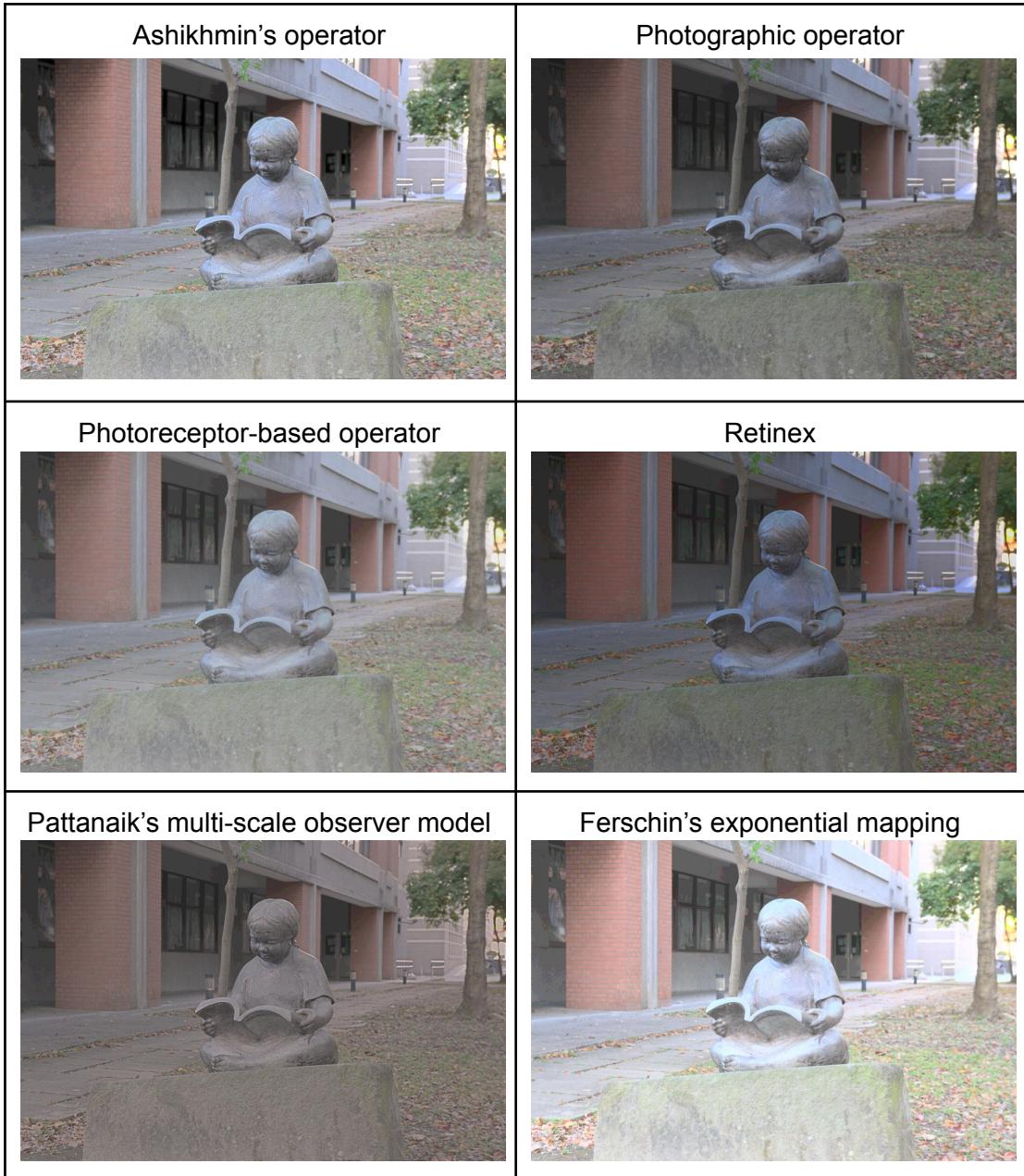
Applying ‘cv2.createTonemapReinhard’, ‘cv2.createTonemapMantiuk’, and ‘cv2.createTonemapDrago’.

我們認為在這三個演算法中，‘Reinhard’ 及 ‘Drago’ 的效果相對還不錯。



- tmo package

這個package 內有24個演算法，我們從中挑選幾張效果不錯的演算法：



- Bilateral Filter

Coding file in 'code/bilateralFilter.py'，可調動參數有：

- compression\_factor (for reducing contrast of large scale)
- sigma\_s, sigma\_r, kernel\_size (for Bilateral Filter algorithm)
- ldr\_scale\_factor (for output image)

我們依照老師上課內容 (lec04\_tonemapping.pdf - p.56) 進行實作，步驟如下：

- 先將前面輸出的 HDR 讀進來，並且算出其 intensity (b, g, r 比例分別為 0.1, 0.65, 0.25)。
- 接著 apply Bilateral Filter algorithm on intensity，得到 large scale 及 detail (intensity / large scale)。

Bilateral Filter algorithm 是照著上課講義所提供的公式進行實作：

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi) g(I(\xi) - I(x)) I(\xi)$$

$$k(x) = \sum_{\xi} \left[ f(x, \xi) g(I(\xi) - I(x)) \right]$$

$$G_{\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- f 為 spatial Gaussian function，此 function 將 target pixel 及其 neighbor pixel 的距離與指定的 spatial sigma 做 Gaussian function。
- g 為 Gaussian function on the intensity difference，將 target pixel 及其 neighbor pixel 的 intensity difference 與指定的 intensity sigma 做 Gaussian function。
- k 為 normalization factor。

- 對 large scale 做 Reduce contrast (large scale \* compression factor)，再乘上原本的 detail 得到新的 intensity。
- 最後根據下圖公式算出圖片的 RGB，再乘上 ldr\_scale\_factor 即得到最終輸出的 image。

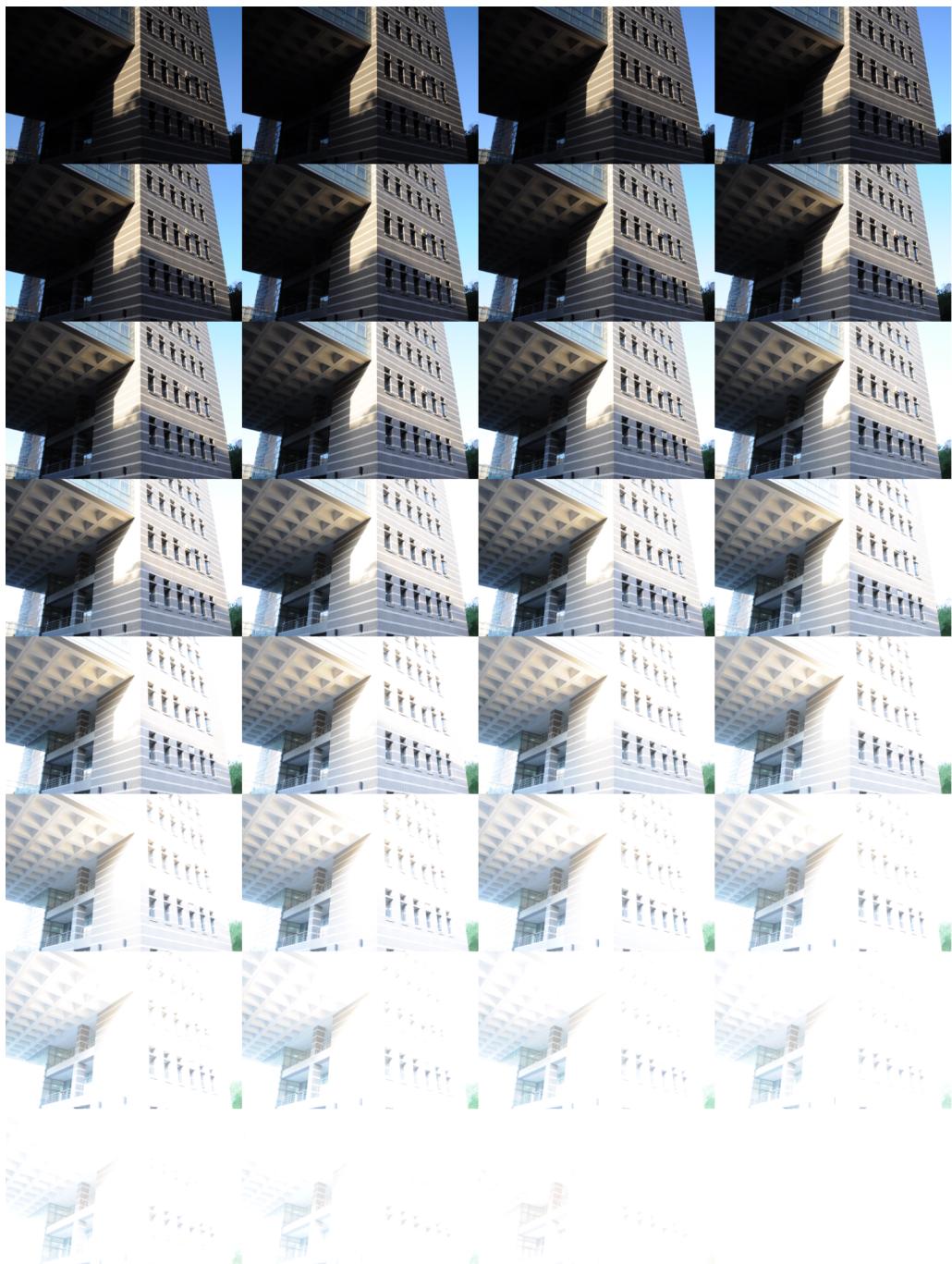
$$\begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} = \begin{bmatrix} L_d \frac{R_w}{L_w} \\ L_d \frac{G_w}{L_w} \\ L_d \frac{B_w}{L_w} \end{bmatrix}$$

output image of Bilateral Filter:

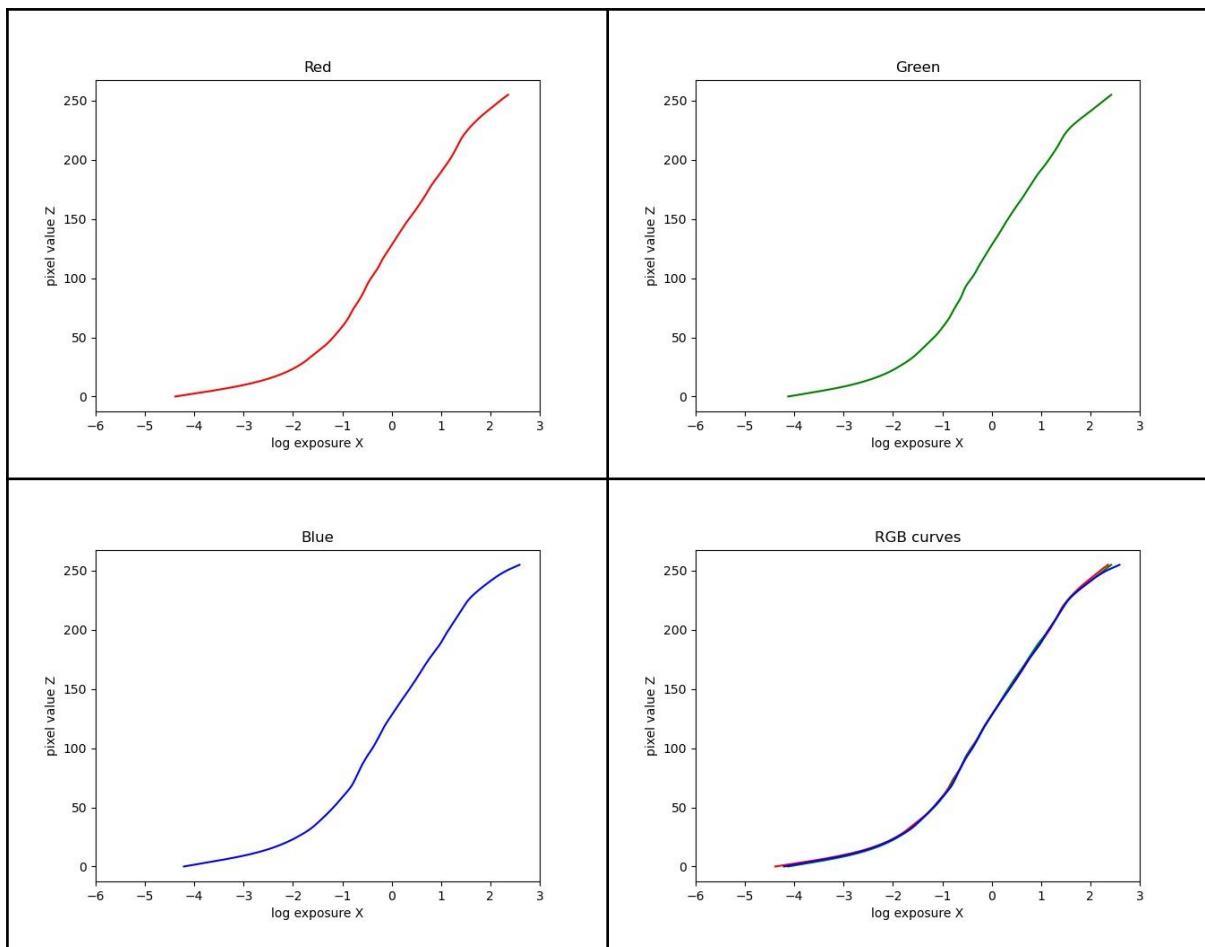


## 5. Appendix

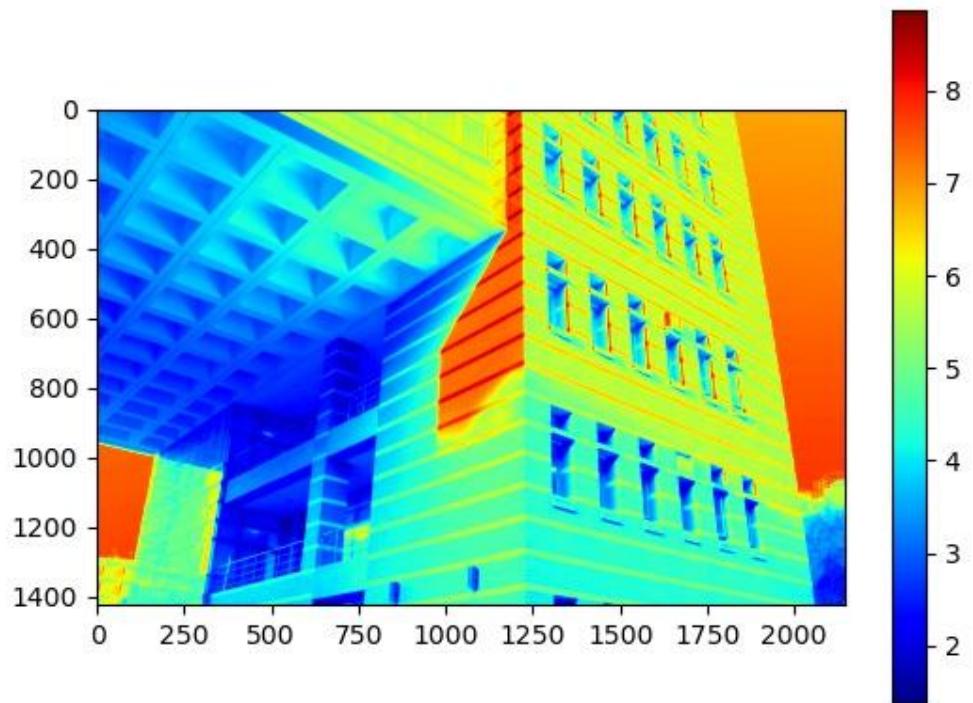
以下展示另一組照片的執行結果。



- Response Curves



- Radiance Map



- Tone mapping

