# VFX hw1

Author: Shannon Lee 李尚倫 studentID: r07921001

Our goal is employing a standard digital camera , and composite HDR images from multiple exposures using the technique pioneered by [1]. In addition, we do an image alignment using the technique pioneered by [2] first.

Content:



## I. Images collecting

**a. Camera configuration:**

```
Dimensions: 6000x4000
Device make: SONY
Device model: SLT-A77V
Color space: RGB
Color profile: sRGB IEC61966-2.1
Focal length: 18mm
Alpha channel: No
Red eye: No
Metering mode: Center-weighted average
F number: f/16
Exposure program: Manual
Exposure time: from 1/3000 to 1/2
```

**b. Two scene 22 exposures —— exposures:**

exposures = [1/3000, 1/2000, 1/1500, 1/1000, 1/750, 1/500, 1/350, 1/250, 1/200, 1/125, 1/90, 1/60, 1/45, 1/30, 1/20, 1/15, 1/10, 1/8, 1/6, 1/4, 1/3, 0.5'']

```
Display numbers: /raw_img/img_shutter_display.txt      # raw
Float numbers: /raw_img/img_shutter_float.txt         # Usable
```

**c. Two scene 22 exposures —— images:**

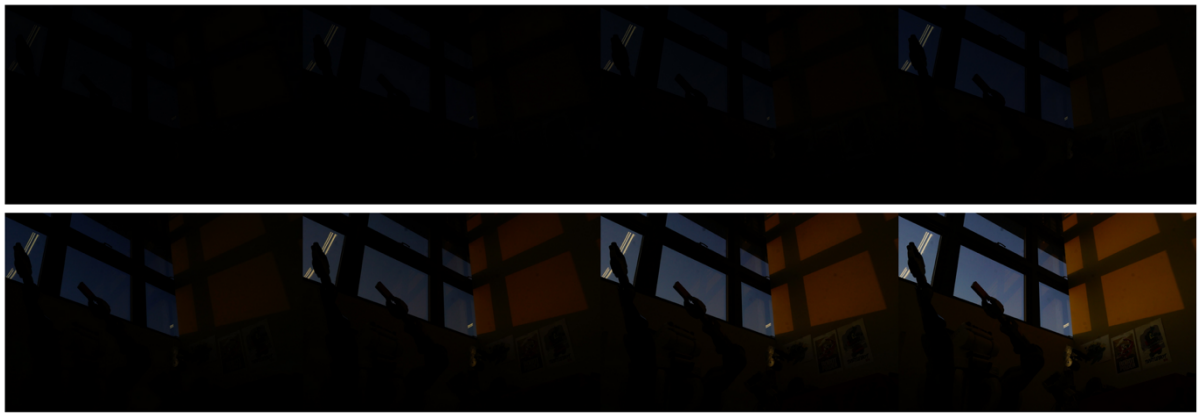
Resized by https://www.iloveimg.com/zh-tw/resize-image/resize-jpg，因為過高的解析度運算時間太長，且可能會因為alignment稍微沒對好就出現干涉條紋。因此拆成1800*1200當實驗結果的高清輸出，900*600當測試，使短時間就可以執行完。

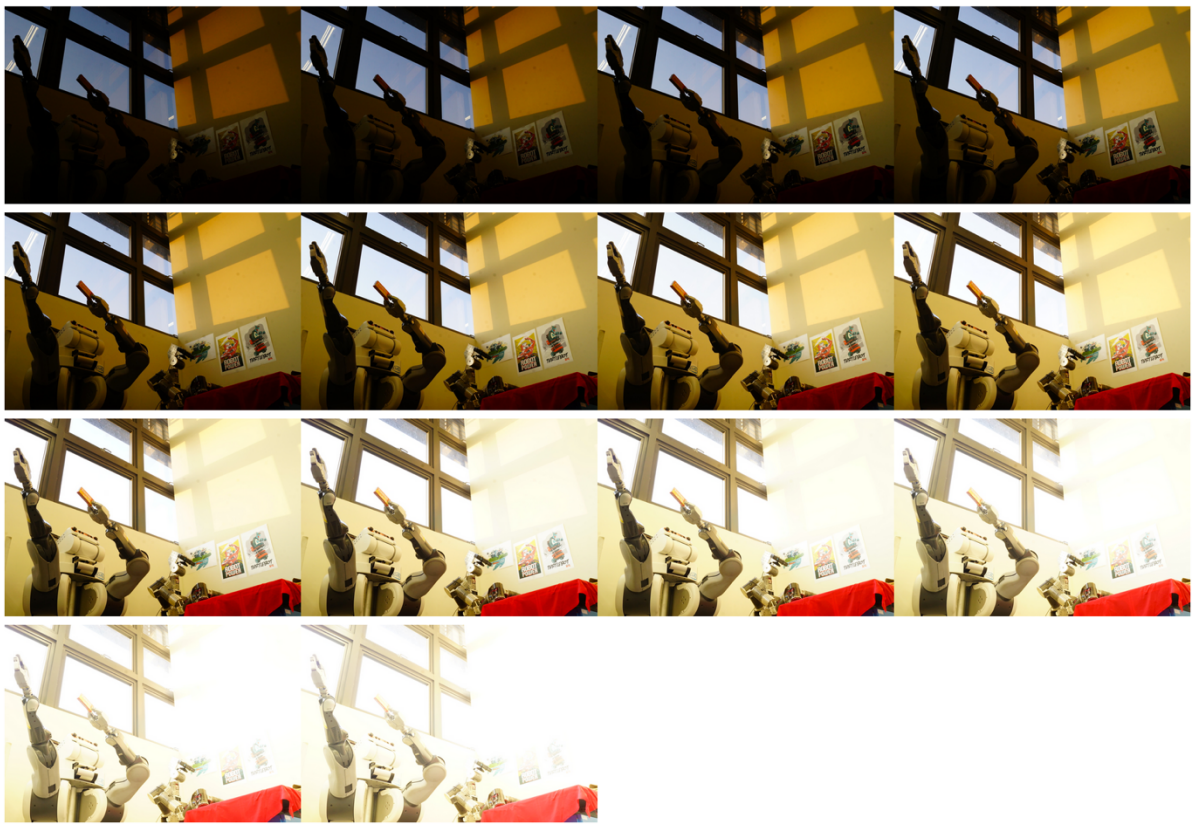
Scene1: Robotics_Corner

```
Raw images 6000*4000: stored at /raw_img/Robotics_Corner    # raw from camera
Use images 1800*1200: stored at /use_img/Robotics_Corner   # used for final production
Test images 900*600: stored at /test_img/Robotics_Corner   # used for quickly test
```

Scene2: Robotics_Corner

```
Raw images 6000*4000: stored at /raw_img/Robot_Power    # raw from camera
Use images 1800*1200: stored at /use_img/Robot_Power   # used for final production
Test images 900*600: stored at /test_img/Robot_Power  # used for quickly test
```





**d. Pick 5 images from each scene:**

因為過多的圖片也無法使 HDR 出來的結果更好，甚至如果過多圖片或挑到很黑或很白的圖都有可能使 alignment 沒做好，合出來的圖片效果更差。因此適當挑選 22 張圖中，從亮處最清楚的圖到暗處最清楚的圖中間的五張圖，如下。

[8, 10, 12, 14, 16] or [3, 5, 9, 13, 20] for robotics_corner


[7,11,13,16,18] for robot_power


## II. Images alignment

參考檔案：src/my_hdr.py

本次作業中總共使用了兩種 alignment 的方式，分別寫成兩個 python function

alignmentImages(imgSet, times, pyramid_level): 為我參考論文[2]後實做的方法。

alignmentImages_cv2(imgSet, times): 為 cv2 的內建 function cv2.createAlignMTB。

cv2 的方法就不詳細解釋，以下介紹我實作[2]的 alignment 演算法流程。

**a. Read 5 different exposure images (line61~67)**

I tried two method. One is only using green channel, and the other one is using gray scale as paper mention:

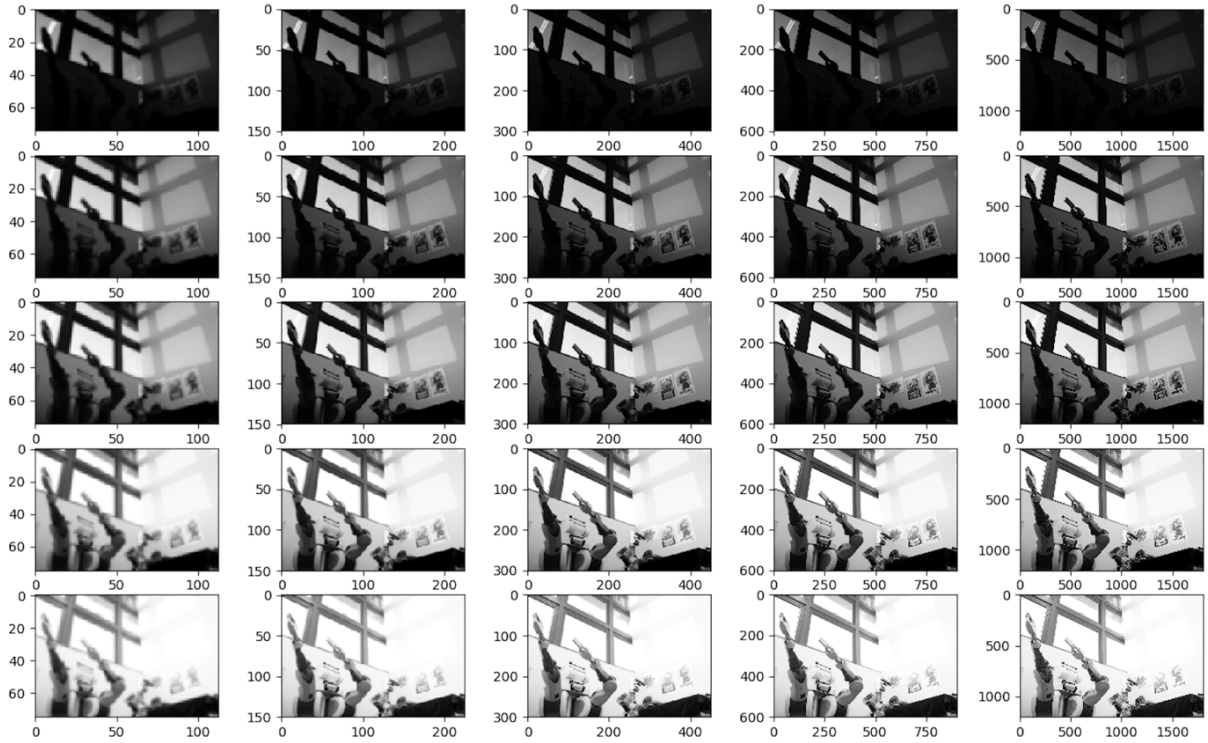
grey = (54*red + 183*green + 19*blue) / 256

At the end, I find that grey does not work better than green, so I use green channel only.

**b. generate image pyramids**

As the paper mentioned, there are a number of ways to go about aligning them, brute force/ gradient/ image pyramid. We choose image pyramid to get the best performance.

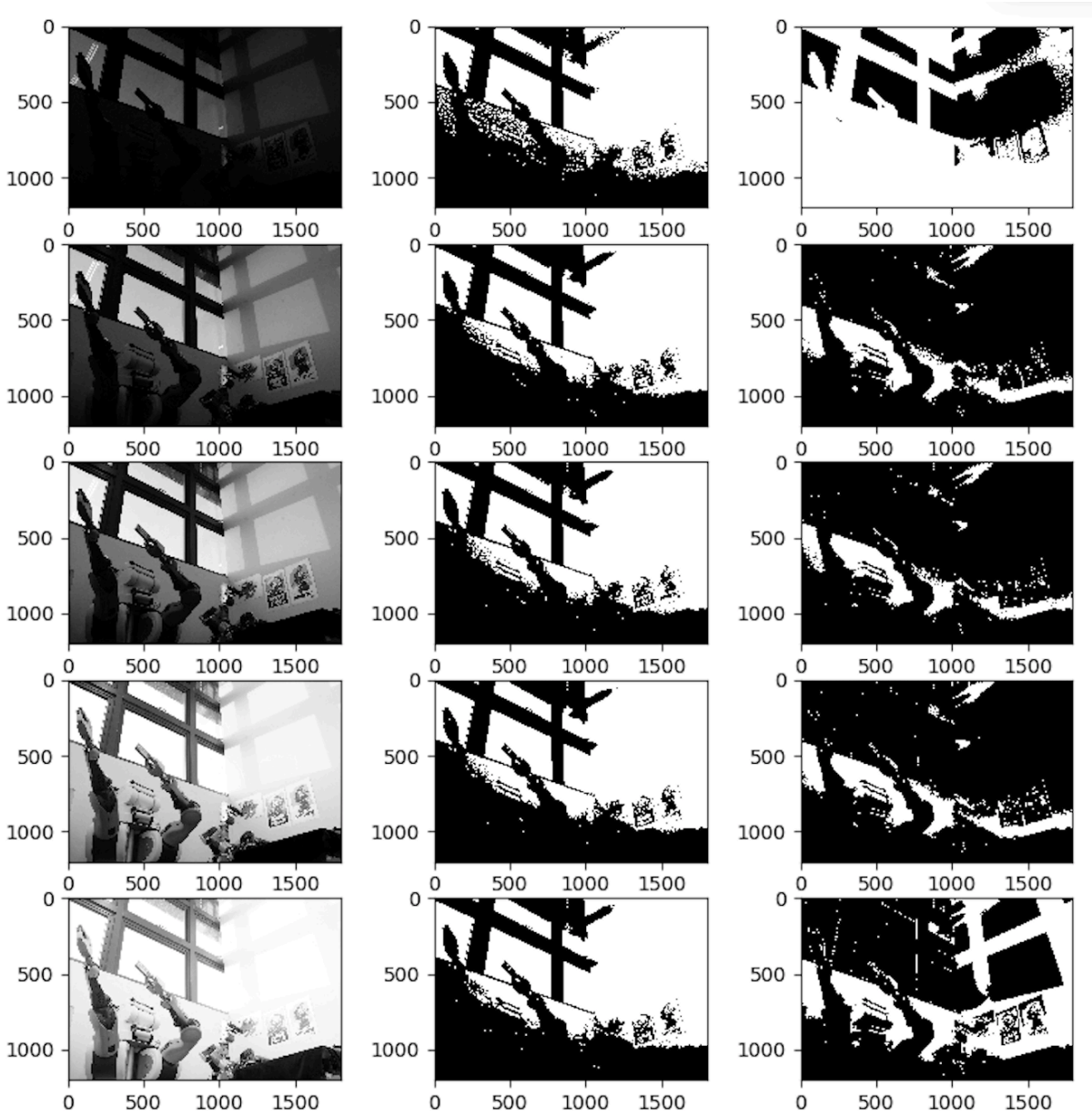
Here I use cv2.pyrDown to build a 5 levels pyramid for each exposure image.
( cv2.pyrDwon is Gaussian pyramid, subsequent images are weighted down using a Gaussian average (Gaussian blur) and scaled down. Each pixel containing a local average that corresponds to a pixel neighborhood on a lower level of the pyramid. )[3]



(如上圖最右側為金字塔的最底層(最大的)，為 1800*1200，第二層為 900*600，第三層為 450*300，第四層為 225*150，第五層為 112*75)

**c. Compute median threshold bitmap(MTB) and exclusive bitmap(EB) for each image at each pyramid level.**

For MTB, I use cv2.threshold to build.
For EB, I use cv2.inRange to build.
（如下圖，以image pyramid最底層為例，左側為原圖(green channel only)，中間為MTB，右側為EB median+-10）



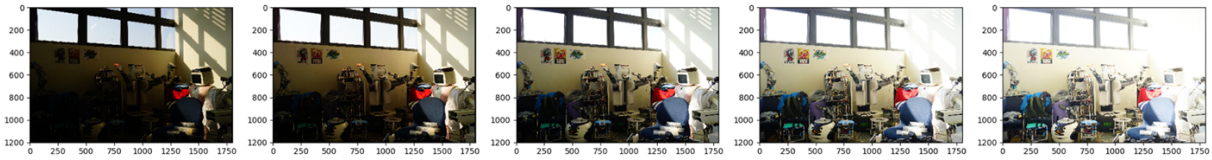
**d. Compare and find the final offset of each image**

I choose middle image(3rd) as reference image, other images would offset to align with it.
We start from highest(smallest) level of pyramid
For each MTB image:

Offset 0~1bit in xy to generate 9 candidates (using cv2.warpAffine)
9 candidates MTB take the XOR with reference image MTB
AND'ing the result with exclusion bitmaps
Sum the result matrix as difference of each candidate with reference
Choose the min difference as the offset, go to next level and multiply this offset by 2

Continues to the lowest level of pyramid, where we get our final offset result to each image.

（如下圖，每張不同目光時間的圖移動的距離 [x, y]pixels 分別由左到右為，
[-4, 0], [-3, -1], [0,0], [0,1], [0,0]，其中中間為[0,0]是因為該圖被定為reference，所以是別人移動來配合它）



## III. Generate HDR image:

參考檔案：src/my_hdr.py

Related function: obtainCRF, generateIE, generateRadianceMap

I implement [2] using obtainCRF and generateIE to recover camera response function(CRF) as known as g function for each channel. Visualize CRF by plotting “Z intensity – log exposure X” graph.

I create radiance map using generateRadianceMap and previous result(g function) to merge five images as one irradiance(E) map. Save as .hdr file and could be visualized by color radiance map.

**a.Input of function**

```
function [g,lE]=gsolve(Z,B,l,w)
```

Z: 我用隨機的方式取了圖片邊界內的 50 個點座標，因此五張照片會有 5*50 的 Z

B: 將曝光時間 T 做 ln

l: lamda，為常數，這裡定為 30

w: 採用簡單的三角形 weighting，z 越接近中間(127)越高，兩側遞減

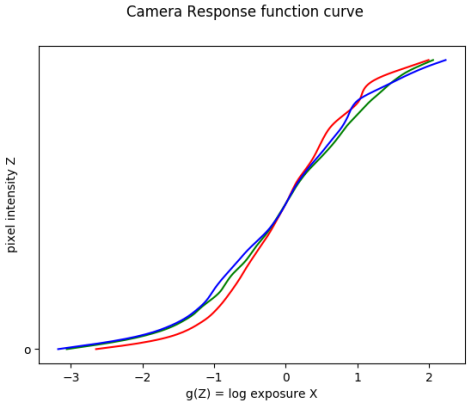
**b. solve Ax=b using least square solution**

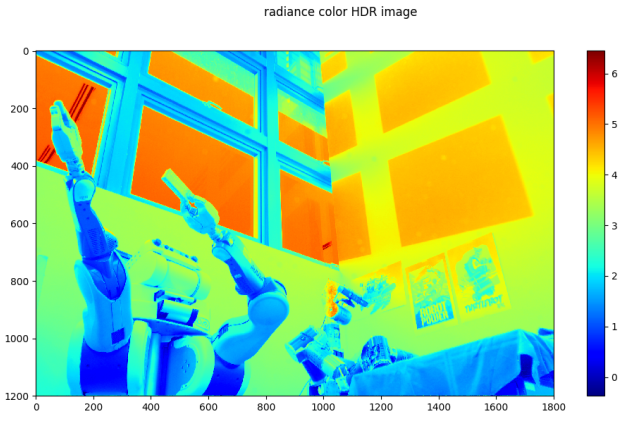
如同[2]提供的 matlab code，將資訊填入 A 矩陣中，最後用 np.linalg.lstsq 來解 least square solution 得 x，x 的前 256 項即為 g.

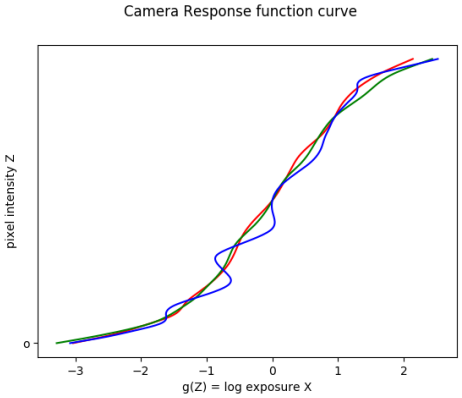
**c. merge images as hdr image**

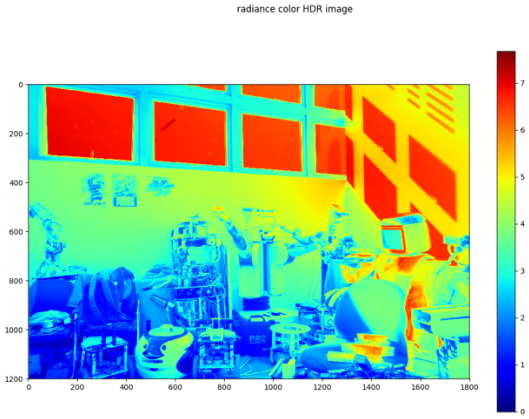
$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})(g(Z_{ij}) - \ln \Delta t_j)}{\sum_{j=1}^{P} w(Z_{ij})}$$

如同[2]提供的公式，將前面得到的 g 和曝光時間，對每個 pixel 還原出該 pixel 的 lnE 值，其中 weight 方法同前面是三角 weighting。最後再 exponential 回去，存成.hdr 檔





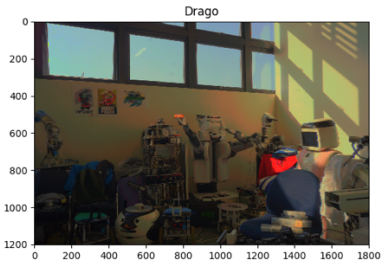



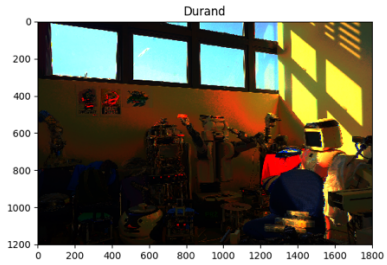

## IV.Tone mapping:

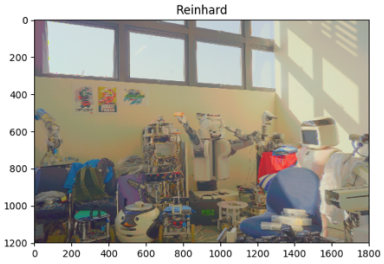
參考檔案：src/my_hdr.py

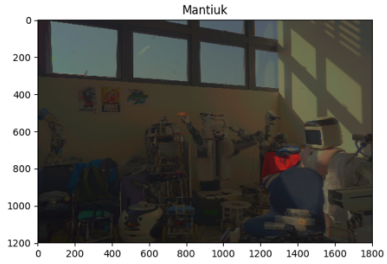
Related function: toneMappingDrago[7], toneMappingDurand[6], toneMappingReinhard[5], toneMappingMantiuk[4]　or Software: Photomatix

這裡主要嘗試了五種方式，依據我上階段生成的.hdr 檔做 tone-mapping，不過就如同老師所言，tone mapping 的演算法每個其實都還有參數可以條，很難鑑別好壞，其中我將調得比較好的幾張放在 VI.Result 裡。

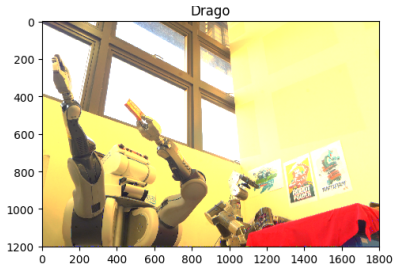


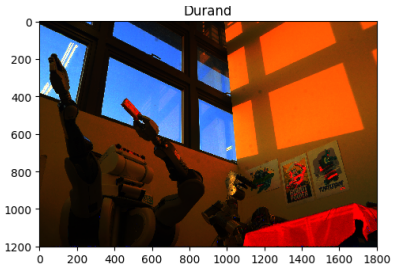



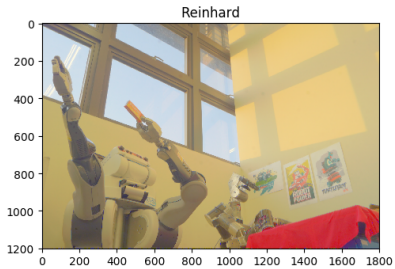



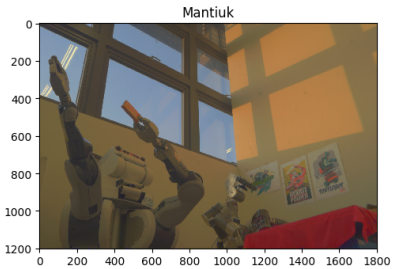








# V. Run the code:

python2 my_hdr.py PHOTO_NAME RESOLUTION ALIGNMENT

PHOTO_NAME: you can choose robot_power or robotics_corner

RESOLUTION: you can choose 1 or 0 for 1800*1200 or 900*600

ALIGNMENT: you can choose 1 or 0 for mine or built-in method

**Example 1:**

python2 my_hdr.py robotics_corner 1 1

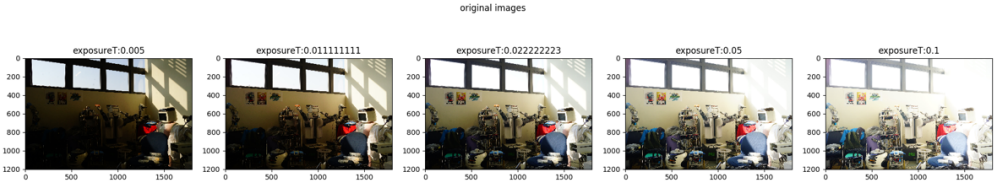
Output:

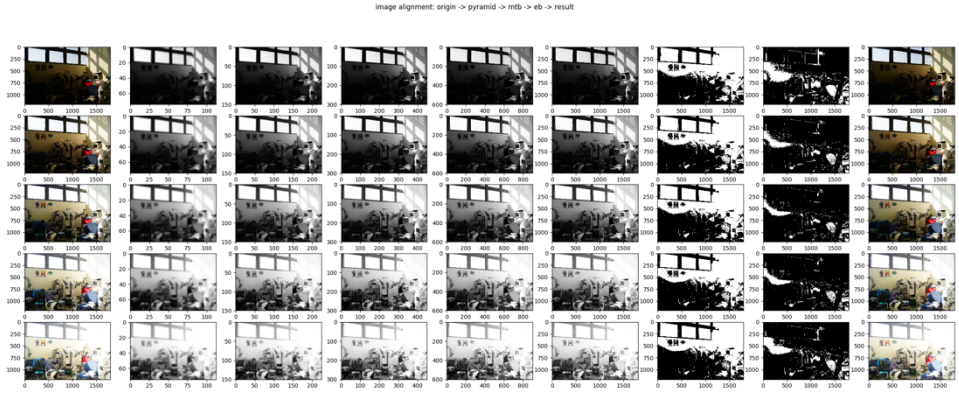
terminal for 5 images offset distance,

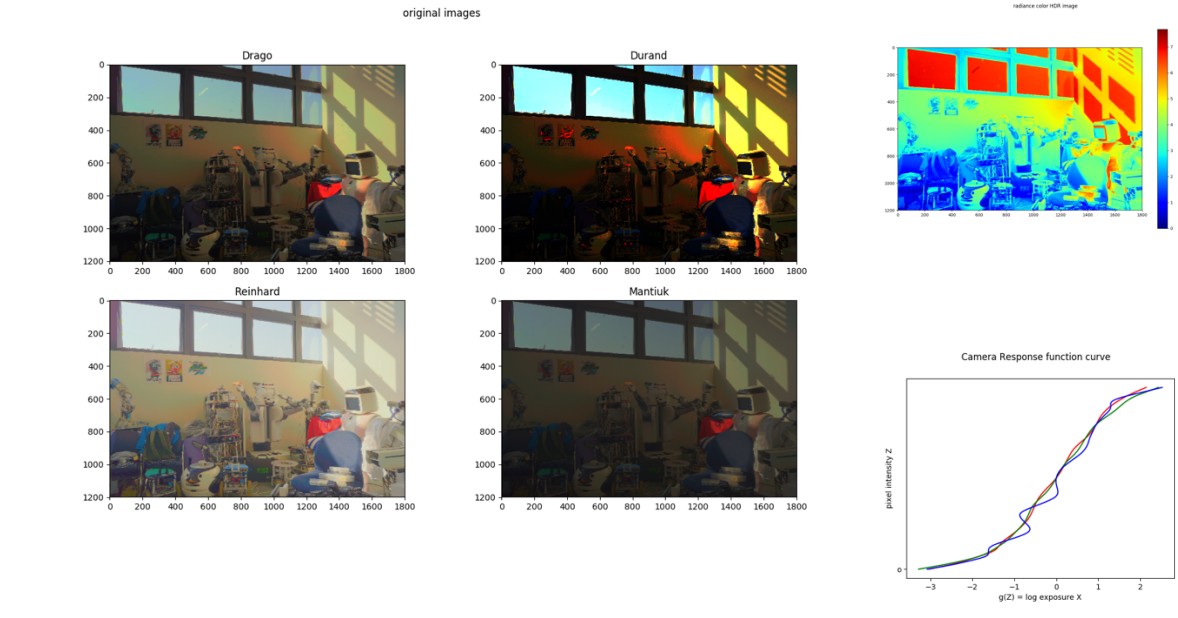
6 matplotlib windows as below,

save Reinhard tonemap result as robotics_corner_test.jpg,

save original radiance map as robotics_corner_test.hdr



**Example 2:**

python2 my_hdr.py robot_power 1 1

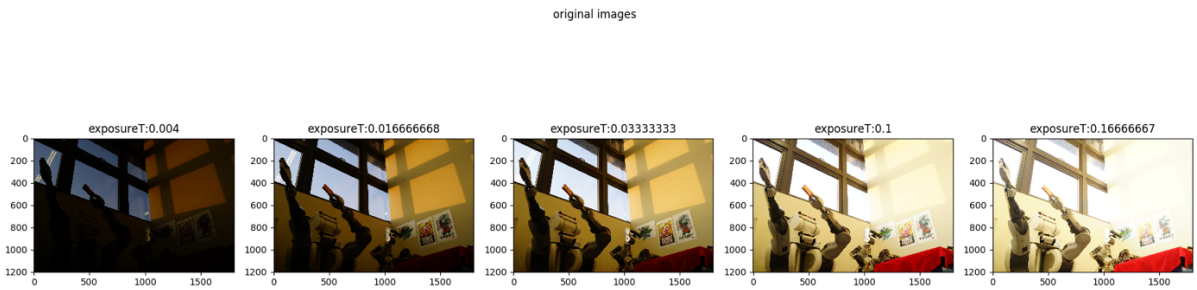
Output:

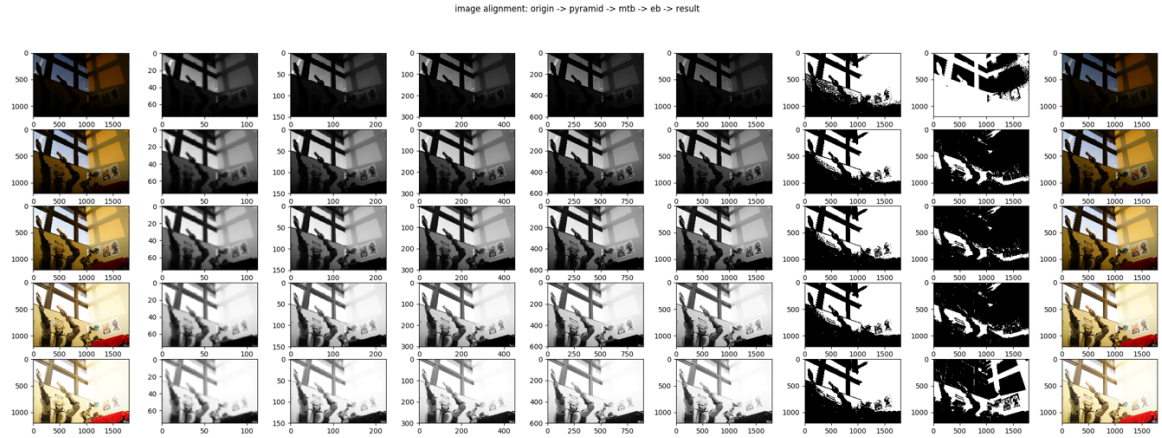
terminal for 5 images offset distance,

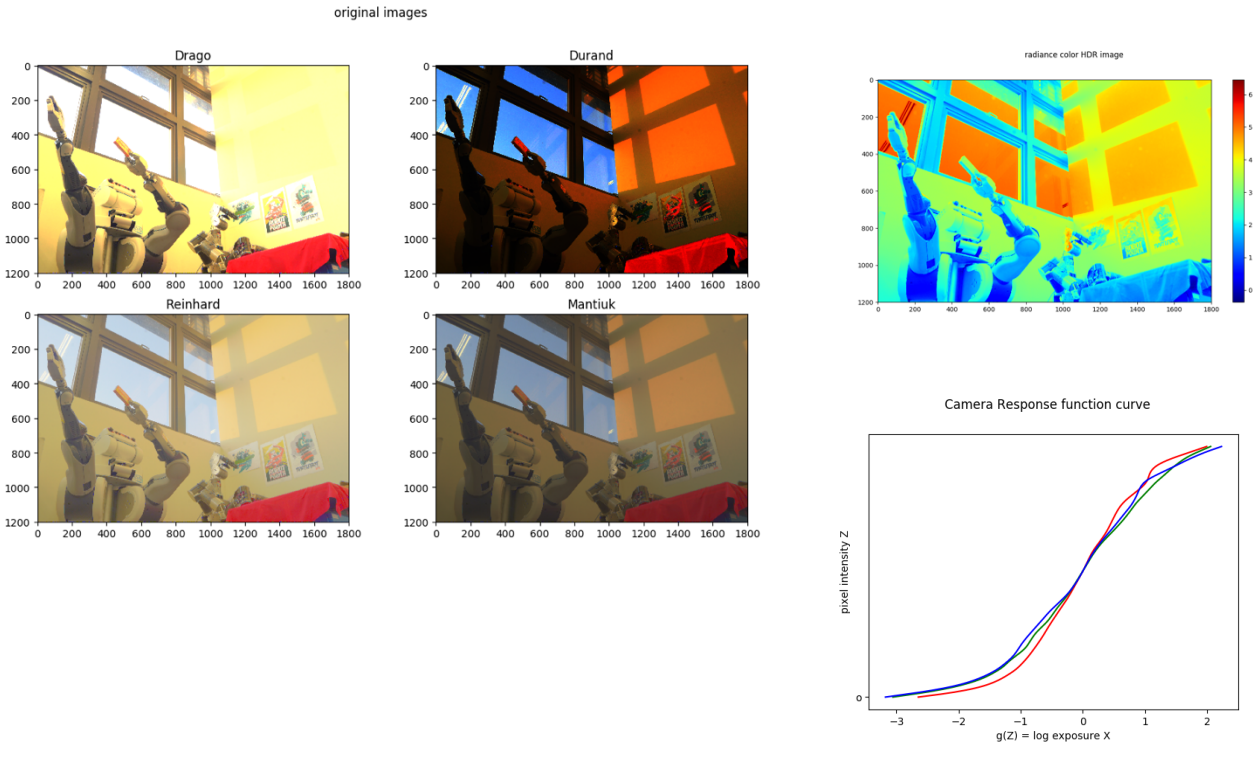
6 matplotlib windows as below,

save Reinhard tonemap result as robot_power_test.jpg,

save original radiance map as robot_power_test.hdr



## VI. Results:


Filename: src/robot_power.jpg Alignment: Myself; HDR(CRF, radiance map): Myself
Tone-mapping: cv2.createTonemapReinhard (based on src/robot_power.hdr)


Filename: src/robotics_corner.jpg Alignment: Myself; HDR(CRF, radiance map): Myself
Tone-mapping: cv2.createTonemapReinhard (based on src/robot_corner.hdr)

## VII. More trials & final artifacts selected:


Filename:
src/robot_power_photomatix.jpg

Alignment:
Myself

HDR(CRF, radiance map):
Myself

Tone-mapping:
Photomatix
(based on src/robot_power.hdr)


Filename:
src/robotics_corner_photomatix.jpg

Alignment:
Myself

HDR(CRF, radiance map):
Myself

Tone-mapping:
Photomatix
(based on src/robotics_corner.hdr)


Filename:
src/robotics_corner2.jpg

Alignment:
cv2.createAlignMTB

HDR(CRF, radiance map):
Myself

Tone-mapping:
cv2.createTonemapDrago
(based on src/robotics_corner2.hdr)


Filename:
src/robotics_corner2_photomatix.jpg

Alignment:
cv2.createAlignMTB

HDR(CRF, radiance map):
Myself

Tone-mapping:
Photomatix
(based on src/robotics_corner2.hdr)

Filename:
src/robotics_corner2_photomatix2.jpg

Alignment:
cv2.createAlignMTB

HDR(CRF, radiance map):
Myself

Tone-mapping:
Photomatix
(based on src/robotics_corner2.hdr)

FINAL SELECTED

## VIII. Source code & images:

https://github.com/shannon112/VFXolo/tree/test/project1_hdrImage

## IX. Reference papers:

[1] Paul E. Debevec, Jitendra Malik, Recovering High Dynamic Range Radiance Maps from Photographs, SIGGRAPH 1997.
[2] Greg Ward, Fast Robust Image Registration for Compositing High Dynamic Range Photographs from Hand-Held Exposures, jgt 2003.
[3] https://en.wikipedia.org/wiki/Pyramid_(image_processing)
[4] Rafal Mantiuk , Karol Myszkowski , Hans-Peter Seidel, A perceptual framework for contrast processing of high dynamic range images, Proceedings of the 2nd symposium on Applied perception in graphics and visualization, August 26-28, 2005, A Coroña, Spain
[5] E. Reinhard and K. Devlin, "Dynamic range reduction inspired by photoreceptor physiology," in IEEE Transactions on Visualization and Computer Graphics, vol. 11, no. 1, pp. 13-24, Jan.-Feb. 2005.
[6] Frédo Durand , Julie Dorsey, Fast bilateral filtering for the display of high-dynamic-range images, ACM Transactions on Graphics (TOG), v.21 n.3, July 2002
[7] Drago, F., Myszkowski, K., Annen, T., and Chiba, N. 2003. Adaptive logarithmic mapping for displaying high contrast scenes. Computer Graphics Forum 22, 3 (Sept.), 419--426.