README.md

VFX Project 1: High Dynamic Range Imaging

Author

- 網媒所 王哲瑋 Che Wei Wang r10944037

Code Structure

- `run.sh` : run `main.py` with predefined arguments.
- `main.py` : will handle command line arguments and then `import lib` and `import utilities` to run the Debevec's HDR Method.
- `lib.py` : implementation of Debevec's Methods.
- `utilities.py` : handle some trivial functions，like `show_multiple_images` .

How To Execute

- Data format:
  - Images should be named from 0 to N, where N is the number of your images.
  - ExposureTime should be stored in a file named exposures.txt , this file is not required if you have exposure time kept in your images' metadata.
- Execution methods:
  i. Execute `./code/run.sh` while in `./code` .
  ii. Run `./code/main.py` with arguments:

Argument	Explanation	Required
input_dir	The directory where you put all your input images	Yes
image_extension	The extension of your image files	Yes
image_num	The number of your input images	Yes
exposure_path	The path to your exposures.txt	Optional
output_dir	The directory where you want to store all the outputs (including g_function plot, hdr file, tone_mapped hdr file with different parameters)	Yes

About The Algorithm

The algorithm being implemented in this project is Debevec's Method。

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \{w(Z_{ij}) [g(Z_{ij}) - \ln E_i - \ln \Delta t_j]\}^2 +$$
$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} [w(z) g''(z)]^2$$

After solving the gunction g in the above formula, I can use the below formula to calculate the radiance of every pixel.

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

This has to be done for every channel: R,G,B. In the end, I will get the radiance map for all three channels, and after stacking them together, I can get the HDR.

Implementation Details

First, I have to handle the reading of the images. My implementation allows user to choose to provide images either with exposure data kept inside the metadata of the images or with an extra file `exposures.txt` that keeps the exposure data.

Then, for sampling pixels, I have tried

- random sampling
- sort the pixels based on their luminosity and then pick pixels uniformly across the sorted list. But the result are almost the same.

Forward, I throw these pixels inside `lib.solve_debevec` to calculate the g function for R,G,B channels: `g_r` , `g_g` , `g_b` , and then derive the radiance map for each channel: `irradiance_r` , `irradiance_g` , `irradiance_b` . By stacking them, I get the HDR image.

To tone-map the HDR image, I implement the Reinhard's Method but only the global operator part.

$$\bar{L}_w = \exp\left(\frac{1}{N} \sum_{x,y} \log(\delta + L_w(x,y)) \right)$$

$$L_m(x,y) = \frac{a}{\bar{L}_w} L_w(x,y)$$

$$L_d(x,y) = \frac{L_m(x,y)\left(1 + \frac{L_m(x,y)}{L_{white}^2(x,y)}\right)}{1 + L_m(x,y)}$$

First I calculate the output tone-mapped luminosity `L_w` using the above three formula and the original HDR luminosity `L_d`.

`key a` and `L_white` are user-defined parameters for different tone-mapped result.

Various combination of `key a` and `L_white` have been experimented with and the result will be presented in later part of the report.

After getting `L_w`, I use the below formula to scale the RGB channels of our HDR image and get the tone-mapped result.

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

Result

Below are the 10 input images with different exposure time.





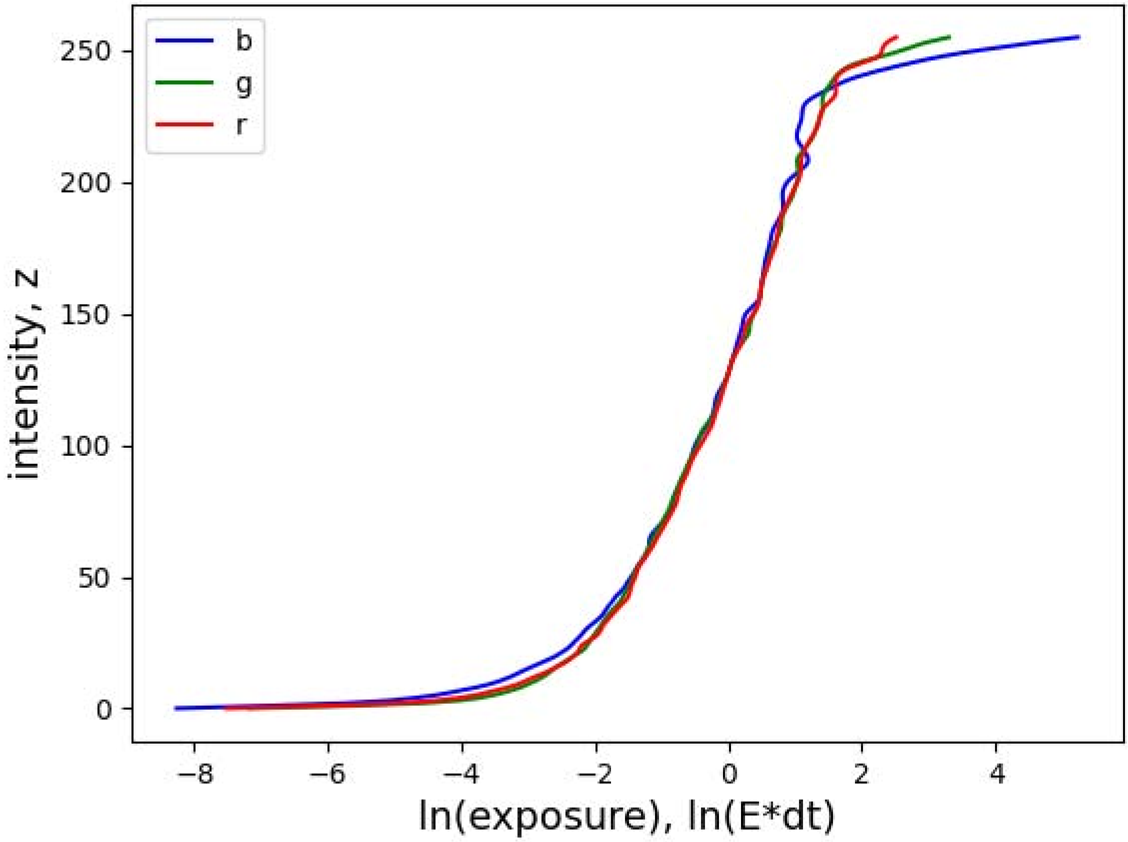





The parameter I choose to use

- lambda = 100
- weighting function : linear-hat

The below plot shows the mapping function g I get for each channel. Can see they are quite matching too each other.

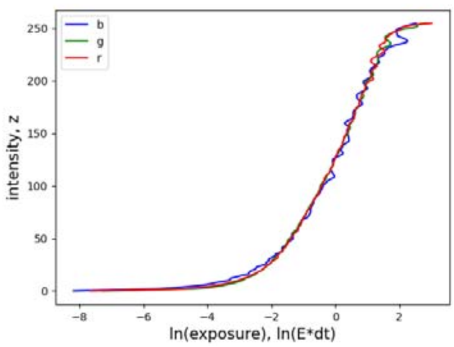


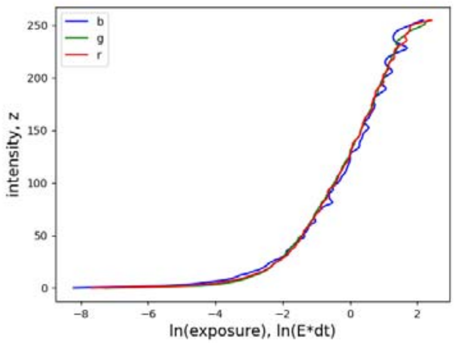
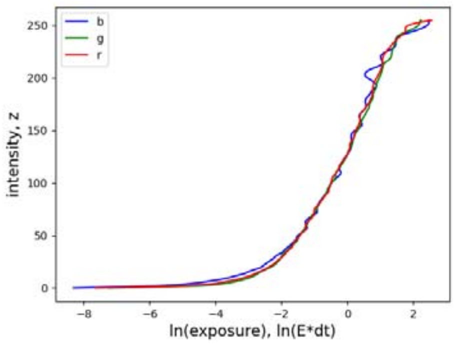
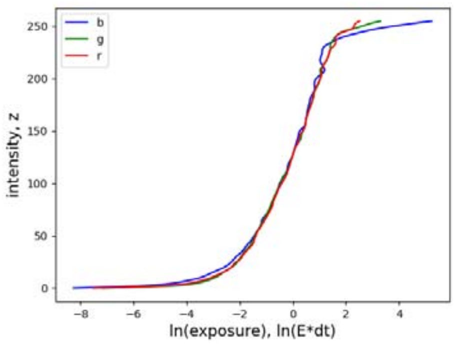
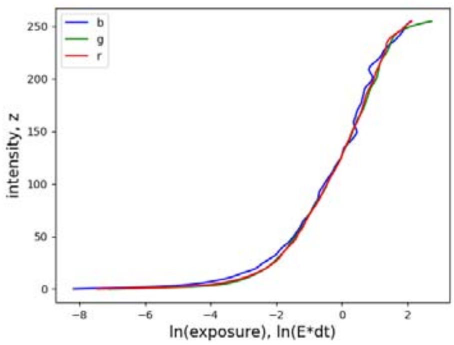
Tone mapped result (Reinhard's Method global operaotr, with a=0.5, L_white=inf) :

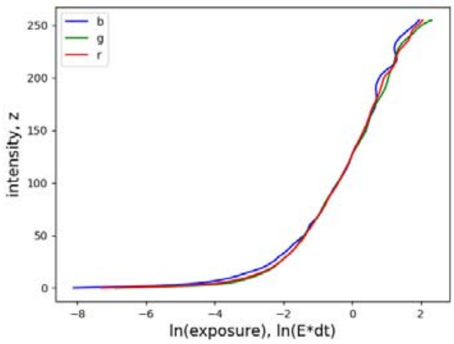
Experiments

Using Different Lambda

To observe the influence of different lambda on the result. I make the tone-mapping parameters(a, L_white) fixed and modify lambda.

Lambda	g	Tone mapped
10		

<table>
<tr><th>Lambda</th><th>g</th><th>Tone mapped</th></tr>
<tr><td>20</td><td>
</td><td></td></tr>
<tr><td>50</td><td>
</td><td></td></tr>
<tr><td>100</td><td>
</td><td></td></tr>
<tr><td>200</td><td>
</td><td></td></tr>
</table>

Lambda	g	Tone mapped
400		

We can see that as lambda goes up, the mapping function g become smoother, but the tone-mapped results have almost no difference.

Using Different Parameters(a, L_white) For Reinhard's Tone-mapping Method

	a=0.18	a=0.3	a=0.4	a=0.5	a=0.75
L_white=0.5					
L_white=1.5					
L_white=3					
L_white=inf