DigiVFX project #1: High Dynamic Range Imaging

R08922079 洪浩翔

About this project

In this project, I implement High Dynamic Range image recovery with the use of raw images to recover radiance map directly, and use tone mapping method from [1] to convert a HDR image back to a normal picture. I take color temperature adjustment [2] and brightness enhancement into account as well after tone mapping, in order to create a more favorable image. As for implementation details, I use Python as my programming language, and use famous packages such as rawpy, PIL and OpenCV to deal with image I/O and processing. For input images, I took 3 pictures with different exposure time, and resized them to smaller size due to memory limitation. Exposure time was recorded in JPEG photo itself and it is hard to read from raw file information, so I provide JPEG files as well to extract exposure time directly. I conduct a series of experiments with different input sets and parameters to determine the best result.

Algorithm

The main algorithms I have implemented are recovering HDR image directly from raw images, and Reinharde's method [1] for tone mapping.

HDR

For HDR image recovery, I use the following equation to recover radiance map from raw input:

$$E_i = \frac{\sum_{j=1}^{P} X_{ij} t_j}{\sum_{j=1}^{P} {t_j}^2}$$

Where i is index of pixel, j is the index of images with different exposures, X is pixel value from each raw image, and t is exposure time of each image.

Tone Mapping

As for tone mapping, the algorithm consists of several parts. The first part is radiance map normalization. I use the formula to normalize each channel of radiance map:

$$normalE_i = \frac{E_i - \min(E)}{\max(E) - \min(E)}$$

Where E indicate the radiance map recovered from previous step. The next part is computing luminance. Because the main purpose of tone mapping is changing the

luminance value of a HDR image to suit the whole HDR image into normal pixel range [0,255], finding the luminance of the image is important. I use the following equation to compute the luminance of the HDR image:

$$L_w = 0.27 \times \mathrm{R} + 0.67 \times \mathrm{G} + 0.06 \times \mathrm{B}$$

Where R, G and B are channels in HDR image. The third part is computing the key value in [1]. The equation is describe below:

$$\mathrm{key} = \exp\left(\frac{1}{N} \times \sum \log(\partial + L_w)\right)$$

where N is the number of pixels, ∂ is a very small number to avoid log(0), and L is luminance from previous step. After the key is computed, scaled luminance and recover steps are as follow:

$$L_i = \frac{a}{key} \times L_{w_i}$$

Where a is a free parameter, and:

$$L_{d_i} = \frac{L_i\left(1 + \frac{L_i}{L_{max}^2}\right)}{1 + L_i}$$

With this equation, the final luminance L_d is created, and thus we can compute the tone mapping output image:

$$\mathrm{I} = \frac{E}{L_w} \times L_d$$

Details

Here are some details about my implementation, including pre-processing, HDR recovery, tone mapping and post-processing.

Pre Processing

I use python package 'rawpy' to deal with my I/O of raw images. However, there is no noise removal on my raw images, and I can't get exposure information from my raw images with rawpy. I use de-noise function in skimage to deal with the noise problem after reading raw images. As for the second problem, I offer JPEG images from my camera as well and use PIL to extract exposure information from JPEG images directly. Because of the memory limitation of my computer, it is impossible to run with original image size. Therefore, I offer a parameter to resize the input images to prevent memory segmentation fault problem.

HDR

In order to run tone mapping with third-party software, I provide radiance map output with '.hdr' format in my program. I follow up the method described in [3] and modify it to a scaled-based version which provide better results.

Tone mapping

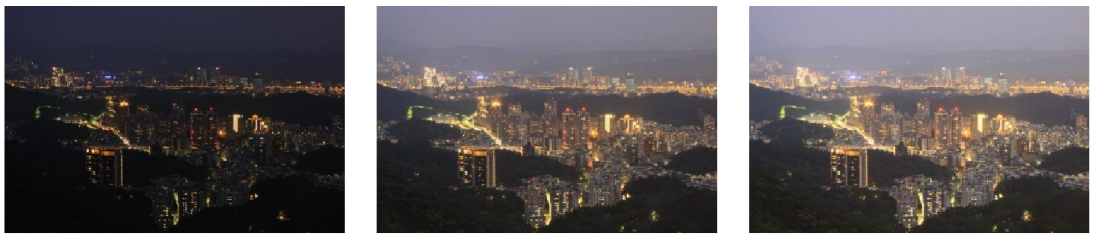
In tone mapping parameters, I set ∂ to 0.0001 to avoid $\log 0$ happened. As for key parameter a, I set it to 0.18 initially as described in [1]. It is a tunable parameter depending on different input images.

Post Processing

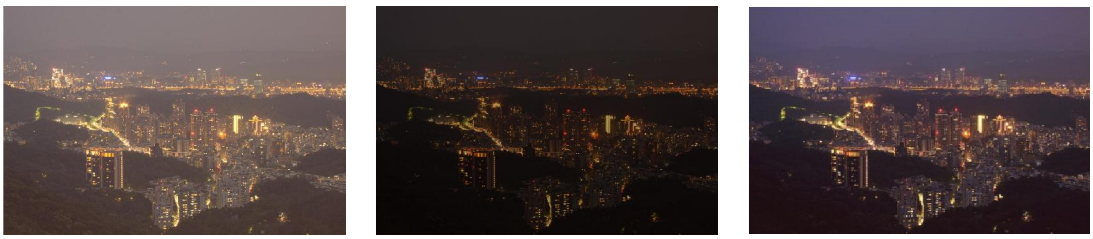
Because I recover HDR image from raw images directly, there is no color temperature information recorded in raw input. As a result, the resultant image would be too red instead of blue displayed in my outputing JPEG image. To solve this problem, I follow the table in [2] to build a color temperature transform function, and provide an iteration parameter to determind the times of transform. Generally, I set initial color temperature transform to 8,000K and the iteration times to 3. Besides, I found that the tone mapping output may be a little dark in my test cases and color temperature transform may unsharpen the output image. Thus, I add brightness enhancement and sharpen functions in my post processing block. I use image enhancement funtion in PIL to tune the brightness, and use filter function to sharpen the output image. Initially, I set brightness tunning factor to 3 and the times of doing sharpening to 3. Both parameters are tunnable in my program.

Results

The following images are input and resultant images I have tried. I run tone mapping not only with my implementation but also with other tone mapping software “tmo”.



Input Images

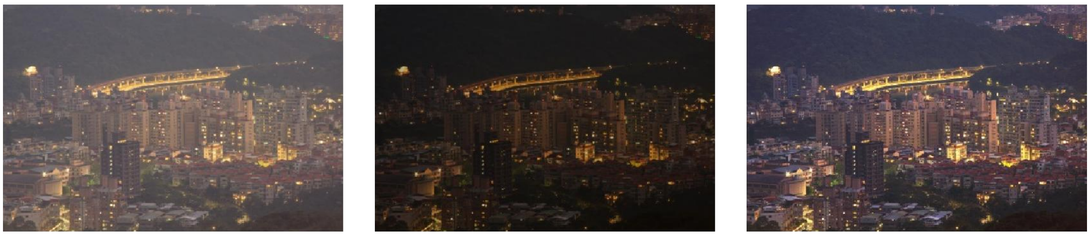


Tone mapping by tmo | My tone mapping result | After post processing

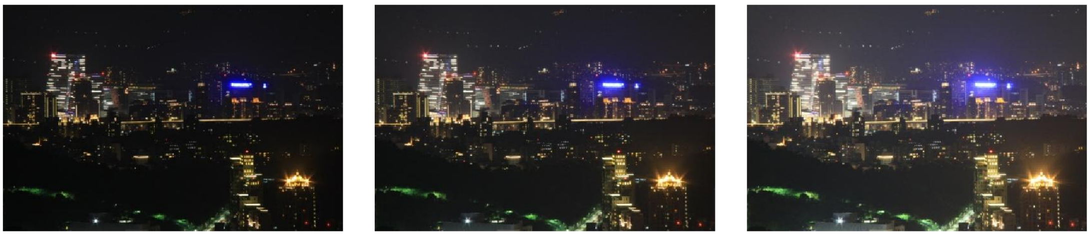
Input Images

Tone mapping by tmo | My tone mapping result | After post processing

Input Images



Tone mapping by tmo | My tone mapping result | After post processing



Input Images

Tone mapping by tmo | My tone mapping result | After post processing

Note: For there is no color temperature info in raw images, color tone of input images and tone mapping outputs may be different.

What I have learned

In this project, I have learned about the basic method for creating HDR image from how to compute radiance map to tone mapping. To handle raw input, this is the first time that I tried to read raw images with python. In order to save the result as .hdr format, I realized the format about a .hdr file and implemented a format writer to deal with it. Besides, I understood the pipline of tone mapping algorithm [1] and the principle of tone mapping. Finally, in order to deal with the lack of raw input images, I realized the methods of de-noising, color temperature transform, brightness enhancement and sharpening. I really learn a lot about digital image in this project.

Files

Main python file [hw1.py, colorTemp.py]
Input image set folders [OurBestInput, intput1, input2, input5]
Output result folders [OurBestOutput, result1, result2, result5]
Third-party tone mapping software [msvcr71.dll, tm_photographic.exe]

Reference

[1] REINHARDE., STARKM., SHIRLEYP. , FERW-ERDAJ.: Photographic tone reproduction for digital images. ACM Trans. Graph.21(3):267–276, July 2002.

[2] http://www.vendian.org/mncharity/dir3/blackbody/

[3] https://m17design.wordpress.com/2005/09/20/hdr-and-rgbe/