HW2 - HDR Imaging and Display

Due Date: 25th, Sep, 2022

Given a sequence with different exposure level (Captured in CUHK-SZ using Sony A7S3, *.ARW *files), merge them into HDR images. Read the **meta data** using *RawPy* to replace the relative parameters of HW01.

You can continue complete your code based on your own HW01 (recommended). If you feel HW01 was not so well completed, just use the reference homework codes instead.

> Experimental Data:
>
> 链接：https://pan.baidu.com/s/1xfBoCOpmvZy0wwuNly6LOw 提取码：0b8w
>
> data.rar

Step 1. Merge LDR RAW into 32-bit HDR Image (50pts)

Since we start from RAW images which are already in linear space, you don't need to estimate camera response curve any more. Convert your RAW data into uint16 and do all the operations with this uint16 data. Create a *HDR_fusion.py* file and code all the relative module inside it.

This step 1 should be performed exactly before the **Demosaicing** process.

> Note: The Lecture06's equations need to calculate the mean pixel value.

Merge LDR images with different exposure levels into an HDR image. Here, you need to define the least square objective function in log-space as given in Lecture06.

```
raw_exposure_fusion() (30pts)
```

Inputs:

1. images # an image list with different exposure levels.
2. weights # the computed gaussian weights given in slides.

Outpus:

1. HDR_image # 32-bits HDR Images

Useful functions to complete: To avoid the error weights caused by this over/under exposured values, you need to compute the mask for each raw images before process the fusion in *raw_exposure_fusion()*.

```
compute_mask() (10pts)
```

Inputs:

1. images # an image list with different exposure levels.
2. low_thres, high_thres # the threshold values for clipping the raw images, here use [0.05, 0.95]*2^14 as default value for experiments. You can also adjust to achieve a better result.

Outputs:

1. mask # a binary mask list for different exposure levels.

Compute the weights for fusion according to the slides.

```
get_fusion_weights() (10pts)
```

Inputs:

1. images # an image list with different exposure levels.

Outputs:

1. weights

Step 2. Demosaic the fused raw data. Save your 32-bit HDR image into 32-bit *.EXR * file. (10pts)\

Process

```
CFA_Interpolation()
```

Implement and process

```
writeEXR()
```

Check your saved results use the give tool-*picturenaut.exe*, or download the linux for mac version as you need.

Step 3. Tone Mapping with Bilateral Filter(30pts)

Now that you have several HDR images, To display them in your 8-bit LCD/LED display, you need to tone map them. For this step, show your results on the report using the fused HDR images from set01.

To realize tone mapping with bilateral filter. Firstly, you need to compute the intensity and then realize a bilateral filter to achieve the base intensity image (Figure 12 in [1]):

```
fastbilateral2d()
```

Inputs:

1. HDR_image_log # 32-bits HDR Image in log space
2. space_sigma = 0.02 * min(width,height)
3. range_sigma = 0.4

Outputs:

1. HDR_image_log_base # 32-bits HDR base Image

Then you need to obtain the details of the HDR image HDR_image_log_detail. Then, the new intensity:

```
compute_new_intensity()
```

Inputs:

1. HDR_image_log_base
2. HDR_image_log_detail
3. gamma # gamma means the compression factor here.

Outputs:

1. LDR_intensity # 8-bits LDR Intensity Image

Finally, combine your new intensity with the color channels and save the results in RGB as a JPG file.

For details, Refer to [1] Fr´edo Durand and Julie Dorsey Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. Siggraph 2002

Step 4. Apply Gamma (10pts)

Even with tone mapping, your images may still appear too dark. In practice, after tone mapping, you still need to apply gamma encoding for images to be displayed correctly. As regular HDTV display devices normally use BT.709 Gamma transfer function, here we gives the BT. 709 Gamma Curve such that you can have a right display on your monitor.

$$V_{709} = \begin{cases} 4.5x; & 0 \le x < 0.018 \\ 1.099x^{0.45} - 0.099; & 0.018 \le x \le 1 \end{cases}$$

Submission

To Grade submission for this homework. We highly recommend that you finish the homework to prevent any issues when you start on the following projects for this class.

Code submission (60%)

The code is required with a simple run, and then the TAs can see the results (in JPG format). Put the data and set the data path as "../data/**" such that can be directly executed.

Results (20%)

The Results shows a good visualization

Report and Description (20%)

For each function you implemented, show the results crops (center 512X512 crops), describe what you have done, and explain your results.

We recommend you write your report using Markdown (like MarkdownPAD2) and export it to HTML format.