CSCI 3290: Computational Imaging and Vision (Due: 23:59, 03 Apr. 2020)

Assignment 3: Tone Mapping

1 Assignment description

Tone mapping is a technique used in image processing to map one set of colors to another to approximate the appearance of high-dynamic-range (HDR) images in a medium that has a more limited dynamic range. Tone mapping operators can be divided into two main types:

- *global operators*: non-linear functions based on the luminance and other global variables of the image.
- *local operators*: the parameters of the non-linear function change in each pixel, according to features extracted from the surrounding parameters.

In this assignment, you'll need to implement these two types of tone mapping with 2 operators respectively: global logarithmic operator and Durand's local operator.

2 Assignment details

2.1 Overview

Given the path of an HDR image, the tone mapping procedure can be described as follows:

```
procedure $ToneMapping(HDR\_Image\_Path,\ ToneMapFunction)$
        1. Load $HDR\_Image$ from $HDR\_Image\_Path$. [provided]
        2. Apply $ToneMapFunction$ to $HDR\_Image$, get tone mapped $LDR\_Image$.
        3. Apply gamma correction to the result $LDR\_Image$. [provided]
        4. Convert the $LDR\_Image$ to 8-bit. [provided]
        5. Save the $LDR\_Image$. [provided]
```

2.2 ToneMap functions

The algorithm of a generic $ToneMapFunction$ can be described as follows:

```
procedure $ToneMapFunction(HDR\_Image)$
        1. Compute the Luminance $L$ of each pixel in the $HDR\_Image$.
           [compute_luminance()]
        2. Apply $ToneMapOperator$ to $L$, compute the Display Luminance $D =$
           $ToneMapOperator(L)$. [tone mapping operators]
        3. Map Display Luminace $D$ on the $HDR\_Image$ to compose the $LDR\_Image$.
           [map_luminance()]
        4. Return $LDR\_Image$.
```

We will discuss the implementation details in next several subsections.

2.3 Operators

2.3.1 Logarithmic operator

We can use the following function to map the luminance $\boldsymbol{L}$ of the HDR_Image to display luminance $\boldsymbol{D}$:

$$D = \frac{\log(L + \tau) - \log\,(L_{min} + \tau)}{\log(L_{max} + \tau) - \log\,(L_{min} + \tau)}$$

where L_{min} and L_{max} are the minimum and maximum luminance of the scene, $\tau = \alpha(L_{max} - L_{min}\,)$, $\alpha \geq 0$. This equation ensures that the maximum and minimum luminance values of the scene are respectively mapped to the maximum and minimum luminance of the display device. Here the range of display luminance is $[0,1]$. Adjusting α will appropriately tune the overall brightness of the reproduced image. In this assignment, we set $\alpha = 0.05$.

This part is related to the function `log_tonemap()`.

2.3.2 Durand's operator

Beside the above two global tone mapping operators, you'll need to implement a local tone mapping operator in this assignment. You'll be implementing a simplified version of Durand's operator. The steps are roughly as follows:

```
procedure $DurandOperator$(Luminace $L$)
        1. Compute the log intensity $\log_{10}(L)$
        2. Filter that with a bilateral filter get the base layer: $BaseLayer =$
           $bilateral\_filter(\log_{10}(L))$
        3. Decompose the detail layer: $DetailLyaer = \log_{10}(L) - BaseLayer$
        4. Compute $\gamma = \frac{\log_{10} contrast}{\max(BaseLayer) - \min(BaseLayer)}$
        5. Reconstruct the luminance: $D' = 10^{(\gamma \times BaseLayer + DetailLayer)}$
        6. Compute the display luminance: $D = D' \times \frac{1}{10^{\max(\gamma \times BaseLayer)}}$
```

where $contrast$ is a user-controllable parameter. We set $contrast = 50$.

The above part is related to the function `durand_tonemap()`.

Since the base layer is computed by the bilateral filter, you may need to implement the filter:

$$I^{\text{filtered}}(x) = \frac{1}{k(x)} \sum_{x_i \in \Omega} f_s(x, x_i;\ \sigma_s) g_r(I(x_i) - I(x);\ \sigma_r) I(x_i)$$

$$k(x) = \sum_{x_i \in \Omega} f_s(x, x_i;\ \sigma_s) g_r(I(x_i) - I(x);\ \sigma_r)$$

where $I^{filtered}$ is the filtered image; I is the original input image; x are the coordinates of the current pixel to be filtered; Ω is the window centered in x, so $x_i \in \Omega$ is another pixel; f_s is the 2D Gaussian kernel (spatial kernel) for smoothing differences in coordinates, here we choose $\sigma_s = 0.02 \times \min(input\_width, input\_height)$; g_r is the 1D Gaussian kernel (range kernel) for smoothing differences in intensities, here we choose $\sigma_r = 0.4$. The spatial kernel size (or the window size) can be computed as $size = 2 \times \max(round(1.5\sigma_s), 1)\ + 1$.

The above part is relate to `bilateral_filter()`. You are allowed to use a bilateral filter from some third-party libraries. In that case, you can just put your function invocation inside the function body of `bilateral_filter()`. If you choose to implement the bilateral filter by yourself, you will get some extra bonus points.

2.4 Mapping Luminance

Since the desired result is an RGB image, we need to map the display luminance ***D*** to the color space. Conventionally, we can scale the RGB values of HDR_Image by $\frac{D}{L}$ to get the LDR_Image:

$$Channel_{input} = Channel_{output} \times \frac{D}{L}$$

where ***L*** is the luminance of the HDR_Image. This part is related to the function `map_luminance()`.

2.5 Summary

In this assignment, you are required to implement tone mapping in **Python 3.4+**. In order to make the skeleton code functional, you need to complete these four functions in the skeleton code: `map_luminance(), bilateral_filter(), log_tonemap(),durand_tonemap().`

The skeleton code depends on **OpenCV** and **Numpy**. You are allowed to import third-party libraries into the code. However, you are **not** allowed to directly use tone mapping operator from these libraries.

Moreover, you can find some HDR images in the `test_images` directory. Tone mapped images will be generated in the `output` directory.

3 Submission guidelines

You need to submit the completed `tone_mapping.py` to the Blackboard.

3.1 Marks

- `map_luminance():      10 points`
- `log_tonemap():        30 points`
- `durand_tonemap():     60 points`
- `bilateral_filter():   extra 20 bonus points if you implement this filter by yourself (with pure Python and Numpy)`

3.2 Put personal information in your source code

In your source code file, type your full name and student ID, just like:

```
#
# CSCI3290 Computational Imaging and Vision *
# --- Declaration --- *
# I declare that the assignment here submitted is original except for source
# material explicitly acknowledged. I also acknowledge that I am aware of
# University policy and regulations on honesty in academic work, and of the
# disciplinary guidelines and procedures applicable to breaches of such policy
# and regulations, as contained in the website
# http://www.cuhk.edu.hk/policy/academichonesty/ *
# Assignment 3
# Name:
# Student ID:
# Email Addr:
#
```

3.3 Late submission penalty

If you submit your solution after the due date, 10 marks will be deducted per day, and the maximal deduction is 30 marks even you delay more than 3 days. But there are hard deadlines as we need time to grade and submit grade. The hard deadline is 29 April 2020. After this day, we will not grade your assignment.