**CSCI 3290: Computational Imaging and Vision**

**2023-24 Second Term**

# Assignment 2: Tone Mapping

Due: 23:59, 18 Mar. 2024. See late penalty policy below.

*PLAGIARISM Penalty: Whole Course Failed.*

## 1 Assignment description

Tone mapping is a technique 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 histogram equalization 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 Histogram equalization operator

The procedure of histogram equalization operator is as follow:

1. Compute histogram $h$ wrt. luminance $I$.
2. Compute the cumulative histogram $c$ wrt. $I$ based on $h$:

$$c(I) = c(I-1) + h(I)$$

3. Normalize $c$ such that its values are in [0, 1].
4. Use the cumulative histogram as tone-mapping function. This could be achieved by either a lookup table or linear interpolation of $c$ between different $I$.

$$D = c(L)$$

As for the number of bins (i.e., intervals) used in histogram, you can experiment it by yourself to get better choice. This part is related to the function `he_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** $DurandOperator(\text{Luminace } L)$

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 $\gamma = \frac{\log_{10} constrast}{\max(BaseLayer) - \min(BaseLayer)}$
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; $\Omega$ 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 points (see 3.2.1).*

## 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 $LDR\_Image$:

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

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 3.6**+. In order to make the skeleton code functional, you need to complete these four functions in the skeleton code: `map_luminance(), bilateral_filter(), he_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 use tone mapping operators 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.
*You will fail the course if you copy others' work, including works from previous years, the Internet etc. Sharing your codes with another student is also prohibited.*

## 3.1 Basic (70 points)

- `map_luminance():` `10 points`

- `he_tonemap(): 20 points`
- `durand_tonemap(): 40 points`

## 3.2 Extra features (30 points)

You should choose one of the following to get the extra 30 points. Please note that if you implement more than one questions, only the one with maximum points will be considered.

### 3.2.1 Implementation of the bilateral filter (30 points)

If you implement `bilateral_filter()` by yourself (with pure Python and NumPy), you will get the following points:

- `implementation of the filter: 20 points`
- `runtime performance: 10 points`

### 3.2.2 Tone mapping of HDR panorama (30 points)

Given a set of HDR images obtained from the same scene, how can we get a tone-mapped panorama?

We have provided a set of input images in the `panorama_extra/inputs` directory. If you choose to complete this question, you need to implement `hdr_panorama.py` with only OpenCV and NumPy and then submit `hdr_panorama.py` and the generated `output_panorama.png` to the Blackboard.

Note that no skeleton codes are provided for this feature, but you may reuse the codes from your Assignment 1 and `tone_mapping.py`. You are allowed to directly use the tone mapping operators from OpenCV for this feature.

Grading criteria:

- `implementation of hdr_panorama.py: 20 points`
- `quality of the generated output: 10 points`

### 3.2.3 Other extra feature related to HDR imaging and tone mapping (30 points)

You are recommended to add extra feature to this assignment: for example, interactive GUI for tone mapping, other tone mapping operators, or obtain your own HDR image with multi-exposure HDR capture, etc. Just use your curiosity and creativity.

If you choose to complete this question, you need to implement the extra feature in a separate `tone_mapping_extra.py` and briefly explain the implemented extra feature in `extra.txt`, and then submit them to the Blackboard.

## 3.3 Put personal information in your source code

In your source code files, 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 2
# Name:
# Student ID:
# Email Addr:
#
```

*Missing of these pieces of essential information will lead to mark deduction.*

## 3.4 Late submission penalty

- **Deduction policy**: 10 marks will be deducted per day.
- **Maximum deduction**: 30 marks even you delay more than 3 days.
- **Hard deadline**: 23:59, May 07 2024. No submission is accepted after the hard deadline.