Homework 5
HDR to LDR Conversion

Student: Yuanyuan Zhao, 2300013077, School of EECS, zhaoyuanyuan@stu.pku.edu.cn
Lecturer: Yisong Chen

Part 1: My `tonemap()`

We designed a custom `tonemap()` function that converts HDR images to LDR with reference to Matlab `tonemap()`. It processes the images by i) Luminance Extraction, ii) Log Compression, iii) Bilateral Filter Decomposition, iv) Base Layer Compression, and v) Linear Luminance Restoration. The specific implementation can be found in the file: myTonemap.mlx.

(a) **Luminance Extraction**

Based on CIE 1931 XYZ color space, we can calculate the luminance with the equation below which reflects human eyes' sensitivity to different colors:

```
L = 0.2126 * R + 0.7152 * G + 0.0722 * B;
```

(b) **Log Compression**

Before further processing, we first compress the pixel values to log field for following reasons:

(i) Normalize luminous variations: The variation scales of pixel values in the dark and bright regions are quite different, for instance, 0.01 vs. 100. Conversion to the log field will amplify the variations in the dark region and compress the bright one, making it easy for the subsequent filter parameter setting.

(ii) Weber-Fechner law: The human eye's perception of luminance is approximately logarithmic sensitive, which corresponds to our log compression.

```
L_log = log10(L + 1e-6);  % add 1e-6 to avoid log(0)
```

(c) **Bilateral Filter Decomposition**

We implemented a `bilateralFilterDecomposition(I, sigma_s, sigma_r)` function to decompose image into base layer and detail layer (`L = baseLayer + detailLayer`) for subsequent base layer compression. Specific implementation can be found in the file: bilateralFilterDecomposition.mlx.

(i) Generate spacial Gaussian kernel with $3 \times sigma\_s$:

```
kernel_radius = ceil(3*sigma_s);
[X, Y] = meshgrid(-kernel_radius:kernel_radius,
        -kernel_radius:kernel_radius);
spatial_kernel = exp(-(X.^2 + Y.^2)/(2*sigma_s^2));
```

(ii) Calculate window and luminous kernel for each pixel:

```
window = I(i_min:i_max, j_min:j_max);
intensity_diff = window - I(i,j);
intensity_kernel = exp(-(intensity_diff.^2)/(2*sigma_r^2));
```

(iii) Combine weights and normalize:

```
total_weight = spatial_kernel(
                    (i_min:i_max)-i+kernel_radius+1,
                    (j_min:j_max)-j+kernel_radius+1)
                .* intensity_kernel;
sum_weight = sum(total_weight(:));
baseLayer(i,j) = sum(window(:) .* total_weight(:))
                / sum_weight;
```

(d) **Base Layer Compression**

It's the key procedure in the `tonemap()`, whose main purpose is to map the wide luminous range in the HDR image to a small one, i.e., [0, 255], to correctly display the image on common devices. We employ the compression method below:

$$L_{compressed} = \log_{10} \frac{10^{baseLayer} \cdot whitePoint}{10^{baseLayer} + whitePoint}$$

so that the *whitePoint* can be interpreted as if the luminance is not compressed, that is, the original linear version of base layer compression:

$$L = \frac{baseLayer \cdot whitePoint}{baseLayer + whitePoint}$$

(e) **Linear Luminance Restoration**

The last step, combining the compressed `baseLayer` and the `detailLayer` and mapping the log field luminance to the original linear filed:

```
L_out = L_compressed + detailLayer;
L_out = 10.^L_out;
```

Part 2: Result Analysis

(a) **Tone-mapped Images**
As shown in Figure 1, 2, 3, the tone-mapped images are much clearer than HDR images. Thus, our custom `tonemap()` is satisfactory to a large extent.

(b) **Matlab `tonemap()`**
But when it comes to the differences between Matlab `tonemap()` and our custom one, it

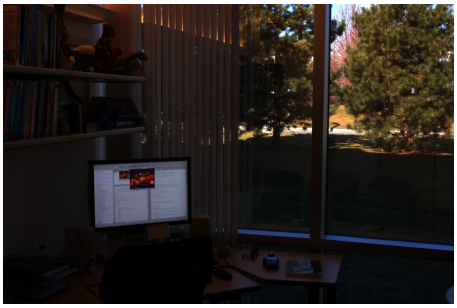
(a) HDR Image

(b) Matlab `tonemap()`

(c) Custom `tonemap()` whitePoint=0.5

(d) Custom `tonemap()` whitePoint=1

Figure 1: before and after `tonemap()`, $\gamma = 2.2, \sigma_{Space} = 10, \sigma_{Intensity} = 0.1$



(a) HDR Image

(b) Matlab `tonemap()`

(c) Custom `tonemap()` whitePoint=0.5

(d) Custom `tonemap()` whitePoint=1

Figure 2: before and after `tonemap()`, $\gamma = 2.2, \sigma_{Space} = 10, \sigma_{Intensity} = 0.1$

(a) HDR Image

(b) Matlab `tonemap()`

(c) Custom `tonemap()` whitePoint=0.5

(d) Custom `tonemap()` whitePoint=1

Figure 3: before and after `tonemap()`, $\gamma = 2.2, \sigma_{Space} = 10, \sigma_{Intensity} = 0.1$

seems that the former makes the images 'whiter'. As to the reason behind this, I suppose it is because of the default parameter settings and varying compression methods.

Part 3: Thoughts

The key insight of tone-map HDR conversion is to decompose basis and details and apply the compression to the former. Essentially, designing such a function is to find out a mapping from a wide luminance range to a small one. The idea of decomposition and compression is worth learning from.