Homework Assignment 2

该作业包含了高动态范围（High Dynamic Range，HDR）成像、噪声校准、颜色较准和色调映射。其中，HDR 成像可将多个不同曝光的图像合并为一个具有高动态范围的图像。噪声校准可找出相机噪点并用于改善 HDR 图像。颜色较准可将图像中的颜色和真实 RGB 值匹配。色调映射可将 HDR 图像的动态范围压缩到 8 位，便于显示。

1. HDR 成像

该部分需要完成图像的线性化（JPEG）、合并不同曝光值的图像和评价生成的 HDR 图像质量。该部分的数据由作业提供，共 2 组图像：JPEG 和 NEF。每组图像包含 16 张不同曝光时间$t^k = \frac{1}{2048} \times 2^{k-1}$的图像，其中$k$为图像的命名编号。

1.1 处理 RAW 图像

下载并安装 dcraw 工具，该工具可在多个平台下处理 RAW 图像。本实验在 Linux 平台下使用如下命令：

dcraw -n 100 -w -o 1 -q 3 -T4 exposure*.nef

该命令行的参数的意义为：

- -n 100：利用微波法消除杂讯的同时保存影响细节。杂讯临界值建议使用 100 至 1000 之间的数值。
- -w：使用相机所指定的白平衡。如果在档案中找不到此项资料，显示警告信息并改用其他方式调整白平衡。
- -o 1：使用 sRGB D65（预设值）。
- -q 3：使用 Adaptive Homogeneity-Directed（AHD）内插法来进行影像的解码。
- -T：输出 TIFF 格式（附元数据）的影像档案。
- -4：输出 16 位元线性档案（固定全白色值，不改变 gamma 值）。

其中，使用-n 100 的原因是在处理较暗的图像时会产生大量噪声，会使后续生成的 HDR 图像也包含大量噪声。因此需要使用该参数去除噪声。

1.2 线性化渲染图像

RAW 图像是线性的，因此不需要线性化即可用于合成 HDR 图像。而 JPEG 图像经过渲染，是非线性的，因此在本节先对其线性化。给定图像k像素点$\{i,j\}$处的像素值：

$$I_{ij}^{k} = f(t^{k}L_{ij})$$

其中L_{ij}为真实辐射值。因此可通过反函数求解真实辐射值。反函数难求，但在场景固定的情况下，可以通过令$g = \log(f^{-1})$并求解最小二乘优化问题：

$$\min_{g,L_{ij}} \sum_{i,j}\sum_{k}\{\omega(I_{ij}^{k})[g(I_{ij}^{k}) - \log(L_{ij}) - \log(t^{k})]\}^{2} + \lambda\sum_{z=0}^{255}\{\omega(z)\nabla^{2}g(z)\}^{2}$$

其中$\nabla^{2}g(z) = g(z-1) - 2g(z) + g(z-1)$，$\omega(I_{ij}^{k})$为保证计算更多的依赖于曝光良好的像素点的权重，共有四种取权重方式：uniform、tent、gaussian、photon。求得g后可以计算得每个像素值对应的该场景下的真实辐射值。可表示为矩阵形式，使用 numpy.linalg.lstsq(A, b)求解g。

$$(Av - b)^{2}$$

$$A = \begin{bmatrix} \vdots & \vdots \\ 0 \dots \omega(I_{ij}^{k}) \dots 0 & -\omega(I_{ij}^{k}) \\ \vdots & \vdots \\ 1 & 1 \\ \vdots & \vdots \\ \lambda\omega(z-1) \quad 2\lambda\omega(z) \quad \lambda\omega(z+1) & 0 \\ \vdots & \vdots \end{bmatrix}$$

$$v = \begin{bmatrix} g \\ \log(L_{ij}) \end{bmatrix}$$

$$b = \begin{bmatrix} \omega(I_{ij}^{k})\log(t^{k}) \\ 0 \end{bmatrix}$$

将四种加权方式（uniform、tent、gaussian、photon）求得的g如图 1 所示。

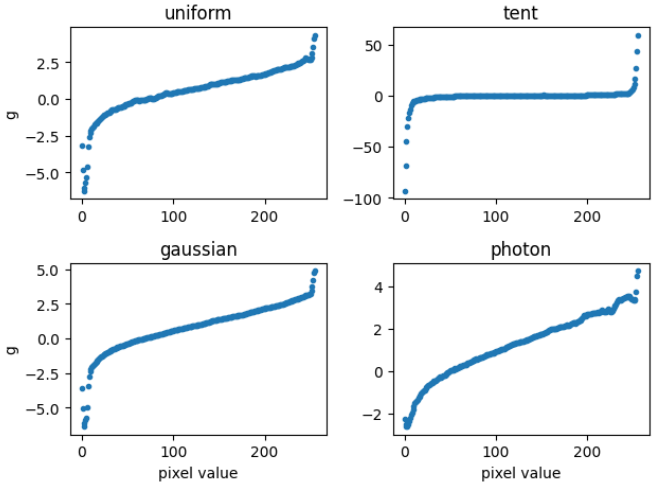

图 1 不同加权方式下的 g 函数值

该部分代码位于./src/linearize.py 中。

1.3 合并不同曝光图像为 HDR 图像

合并方式分为 linear merging 和 logarithmic merging 两种，前者是按照物理精度计算的，后者是由人类视觉感知启发的。

Linear merging 的合并方式：

$$I_{ij,HDR} = \frac{\sum_k \frac{\omega(I_{ij,LDR}^k) I_{ij,lin}^k}{t^k}}{\sum_k \omega(I_{ij,LDR}^k)}$$

Logarithmic merging 的合并方式：

$$I_{ij,HDR} = \exp\left(\frac{\sum_k \omega(I_{ij,LDR}^k)(logI_{ij,lin}^k - logt^k)}{\sum_k \omega(I_{ij,LDR}^k)}\right)$$

其中，$I_{ij,LDR}^k$为原始图像的像素值，$I_{ij,lin}^k$为线性化后的像素值。

我们共有 2 组图片，4 种加权方式以及 2 种合并方式。因此我们可以得到 16 张 HDR 图像，并保存为 exr 格式，存放在./result/文件夹中。

该部分代码位于./src/merge_HDR.py 中。

1.4 评估 HDR 图像质量

我们可以通过图片中的 color checker 来观察颜色的对数值是否成线性变化，来判断 HDR 图像的质量好坏。本节取 color checker 中{4,8,12,16,20,24}号 patch 来评估。

- 首先，通过 matplotlib.pyplot.ginput 函数在每个 patch 中取 2 个点，截一个正方形。这些点保存于./result/pts_jpg.csv 和./result/pts_tiff.csv 中。
- 然后求出每个 patch 对应正方形的 Y 通道均值，并求出对数值。
- 对一组 patch 的对数值求线性回归，可以得到拟合参数和最小二乘误差。

最小二乘误差越小，说明该组 patch 对数值越接近线性关系，HDR 图像质量越佳。求得的最小二乘误差如表 1 所示。具体的拟合参数和最小二乘误差存放在./result/evaluate.csv 中。

表 1 最小二乘误差

原图格式	加权方式	合并方式	最小二乘误差
tiff	uniform	linear	0.0831
tiff	uniform	logarithmic	0.0087
tiff	tent	linear	0.0747
tiff	tent	logarithmic	0.0112
tiff	gaussian	linear	0.0764
tiff	gaussian	logarithmic	0.0104
tiff	photon	linear	0.0147

tiff	photon	logarithmic	0.0164
jpg	uniform	linear	0.0088
jpg	**uniform**	**logarithmic**	**0.0082**
jpg	tent	linear	0.0393
jpg	tent	logarithmic	0.0757
jpg	gaussian	linear	0.0114
jpg	gaussian	logarithmic	0.0114
jpg	photon	linear	0.0138
jpg	photon	logarithmic	0.0140

从表 1 中可以看出采用 jpg 原图、uniform 加权、logarithmic 合并的 HDR 图像是最好的。后续实验也将采用该组合的 HDR 图像。

将这些 HDR 图像的上述 patch 的 log 值绘制成对数图，如图 2 所示。可以看出原图为 tiff 的 HDR 图像都分布在图 2 上侧，分布较为集中且平均亮度较高。相反，原图为 jpg 的 HDR 图像都分布在图 2 下侧，分布较为分散且平均亮度较低。这可能是由于 dcraw 处理较暗 RAW 图时会得到亮度高于 jpg 的图像。

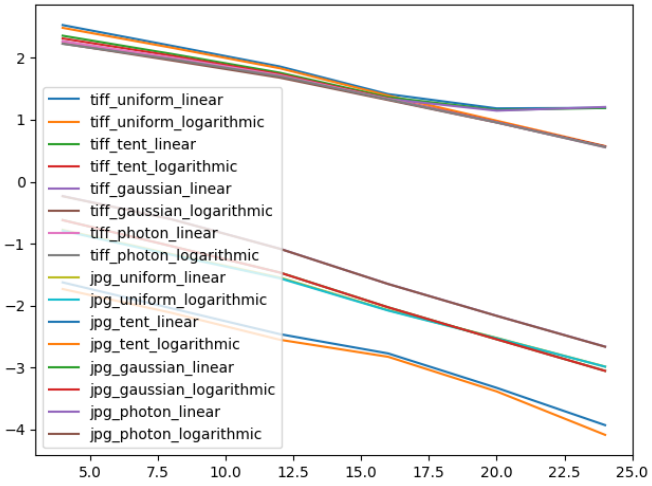

图 2 不同 HDR 图像的 log 值

该部分代码位于./src/evaluation.py 中。

2. 颜色校正和白平衡

我们可以通过 color checker 来校正颜色。首先我们应当获取 color checker 在 D65 照明环境下的标准值，该值由./src/cp_hw2.py 中的 read_colorchecker_gm 函数提供。

颜色校正：先如 1.4 中对 color checker 的 24 个 patch 分别求出 RGB 每个通道的均值。这些点保存于./result/pts_jpg_all.csv 和./result/pts_tiff_all.csv 中。然后通过最小二乘问题求解 HDR 图像的 color checker RGB 坐标到标准 color checker RGB 坐标的仿射变换。然后将该仿射变换应用到整个 HDR 图像中。

白平衡：将每个通道乘以一个值，使得 patch 4 的 RGB 坐标和标准 RGB 坐标完全相等。

校正前后图像如图 3 所示，在展示时使用第 3 部分的色调映射（$K = 0.15, B = 0.95$）。可以看出颜色校正和白平衡后色调过暖的问题得到了解决。校正结果保存为 exr 格式，存放在./result/ccwb/文件夹中。

图 3 颜色校正和白平衡

该部分代码位于./src/color_correction.py 中。

3. 色调映射

HDR 图像的动态范围远大于显示器能显示的动态范围。为了在 8 位的显示器上显示 HDR 图像，需要对 HDR 图像进行色调映射。本节采用色调映射方式有两个参数：参数 K 决定色调映射的亮暗程度；参数 B 用于抑制对比度。

实际色调映射方式有两种：

- 对 RGB 三个通道进行色调映射。
- 将图像从 RGB 域转为 XYZ 域，再转为 xyY 域，对 Y 通道进行色调映射，然后转换为 RGB 域。

实际上在色调映射后，仍需要应用 gamma encoding 才能正确显示图像。

尝试的不同 K 值的结果如图 4 所示。可以看出 K 值越大图像越亮。第一行为直接对 RGB 进行映射，第二行为对 Y 进行映射。

尝试不同 B 值的结果如图 5 所示。第一行为直接对 RGB 进行映射，第二行为对 Y 进行映射。

该部分代码位于./src/photographic_tonemapping.py 中。

图 4 色调映射随 K 值的变化

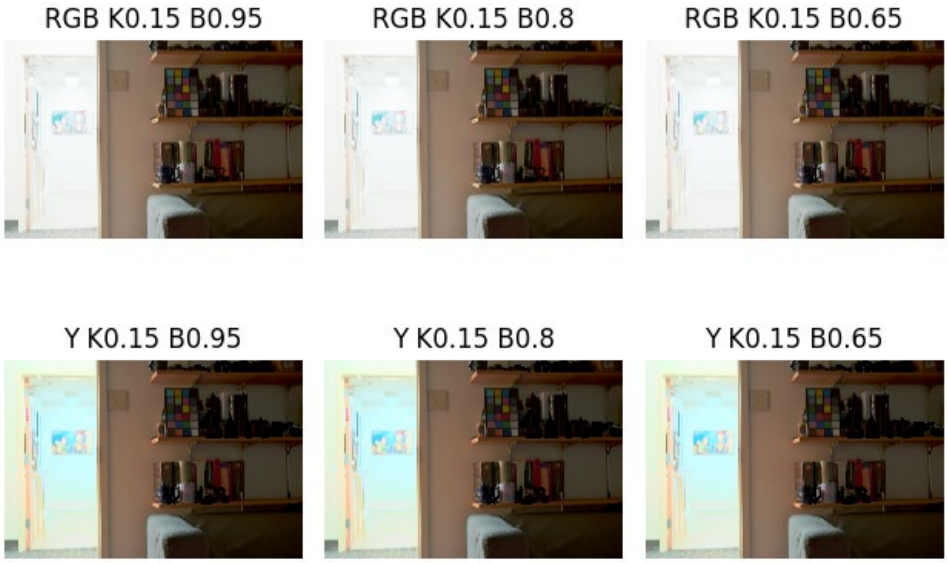

图 5 色调映射随 B 值的变化

4. 尝试自己的 HDR 图像

本节从网上下载两组不同曝光的图像，图像保存在./data/own/文件夹中。每组图像包含 JPEG 格式和 RAW 格式，每种格式 5 张。这两组图像都是由 CANNO D3000 拍摄，曝光时间分别为$\{0.125, 0.25, 0.5, 1, 2\}$。图 6 展示了第一组图像曝光时间最短和最长的图像，图 8 展示了第二组图像曝光时间最短和最长的图像，对两组图像都进行上述 1、3 部分的操作，生成的 HDR 图像保存在./result_own 中。如上文一样采用 tiff 原图、uniform 加权、logarithmic 合并的 HDR 图像进行展示。色调映射参数为$K = 0.15, B = 0.95$。第一组图像如图 7 所示，第二组图像如图 9 所示。

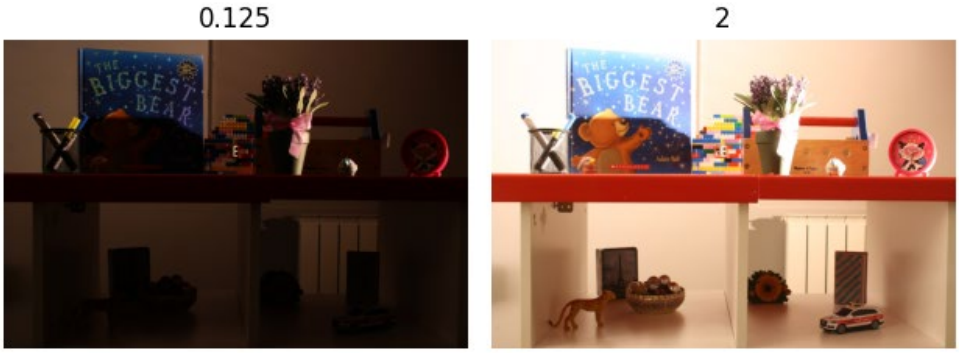

图 6 第一组最短和最长曝光时间图像

图 7 第一组 HDR 成像结果

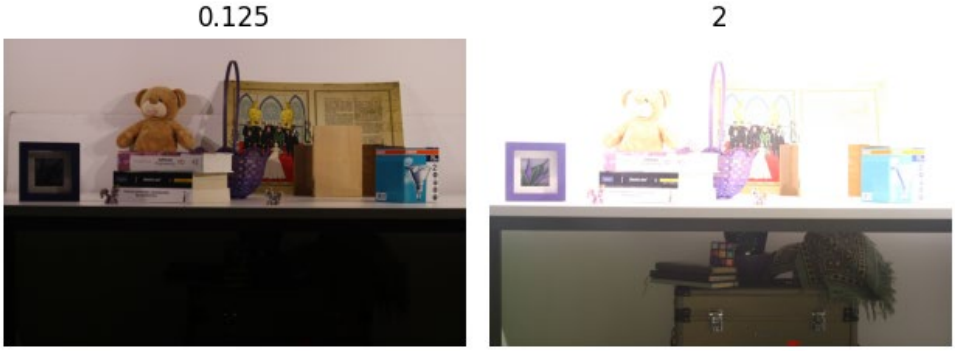

图 8 第二组最短和最长曝光时间图像

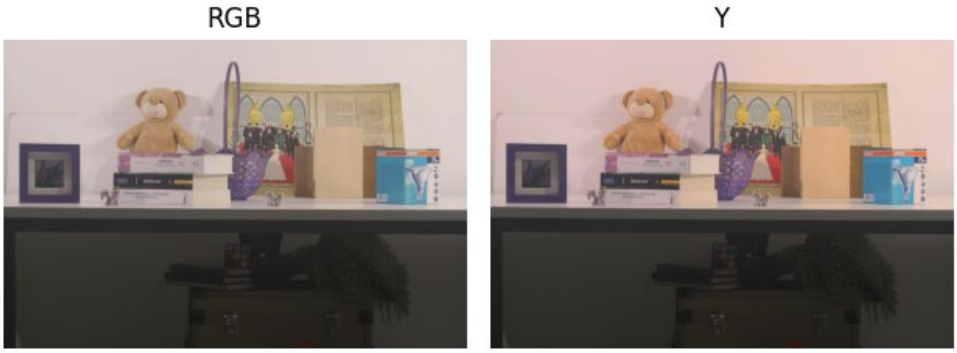

图 9 第二组 HDR 成像结果