# CSC2529-hw4

1010171181 Xinran Zhang

October 2023

## 1 Task1

### 1.1 Fusion weights

First of all, I applied an inverse gamma curve to linearize the images. For each linear image, I calculated the weight of each pixel following the formula and plotted the weights. The results are shown in figure 1.

$$W_{k,i,j} = exp(-4\frac{(I_{k,i,j} - 0.5)^2}{0.5^2}) \tag{1}$$

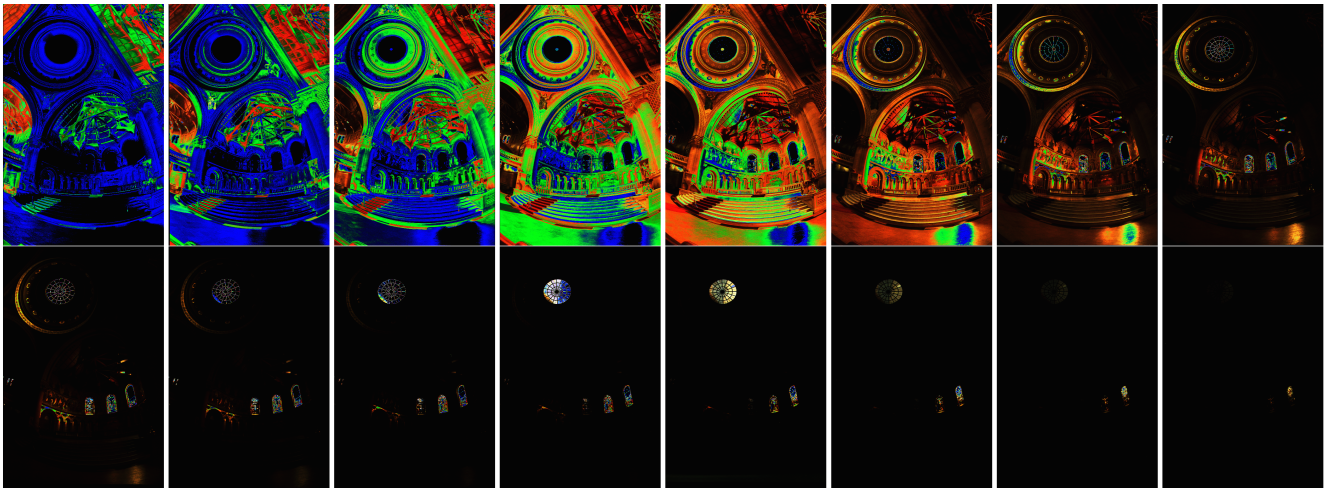

Figure 1: Weights using Debevec' s method (all 3 color channels)

### 1.2 fuse all exposures into a single image

I calculated the HDR according to the following formula, adding to the HDR and scale each time an image was read, and finally calculating the exponential value.

$$\hat{X} = exp(\frac{\sum_k w_k(log(I_{lin_k}) - log(t_k))}{\sum_k w_k}) \tag{2}$$

#### 1.2.1 find a good scale s and gamma parameter $\gamma$

In this part, I use different pairs of s and $\gamma$ to show the normalized HDR image $I_{HDR}$. I chose 3 values for s $\in \{0.1, 0.5, 0.8\}$ , and 3 values for $\gamma \in \{\frac{1}{1}, \frac{1}{2.2}, \frac{1}{5}\}$. Here are the precessed images in figure 2.

$$I_{HDR} = (s \times I_{HDR_{linear}})^{\gamma} \tag{3}$$

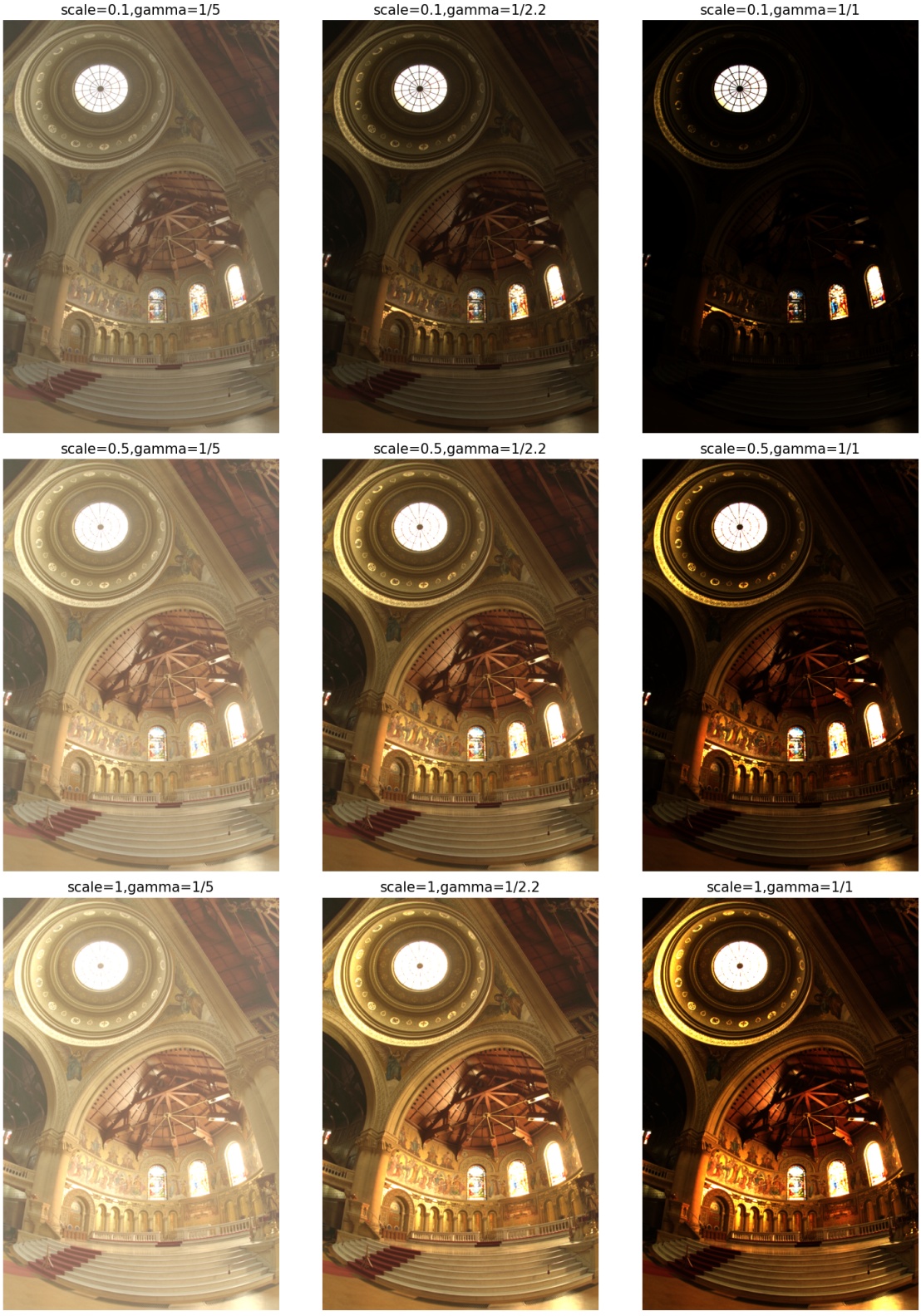


Figure 2: resulting images with different pairs of (s, $\gamma$)

From these resulting images, I would say the best pair of (s,$\gamma$) is (1, $\frac{1}{2.2}$).

#### 1.2.2 openCV's built in functions

The resulting image is shown as figure 3. We can observe that this method actually surpasses the previous one. The resulting picture presents all the details in a manner that should be notably superior to any of the individual images, even when using the 'best pair' of (s, $\gamma$).

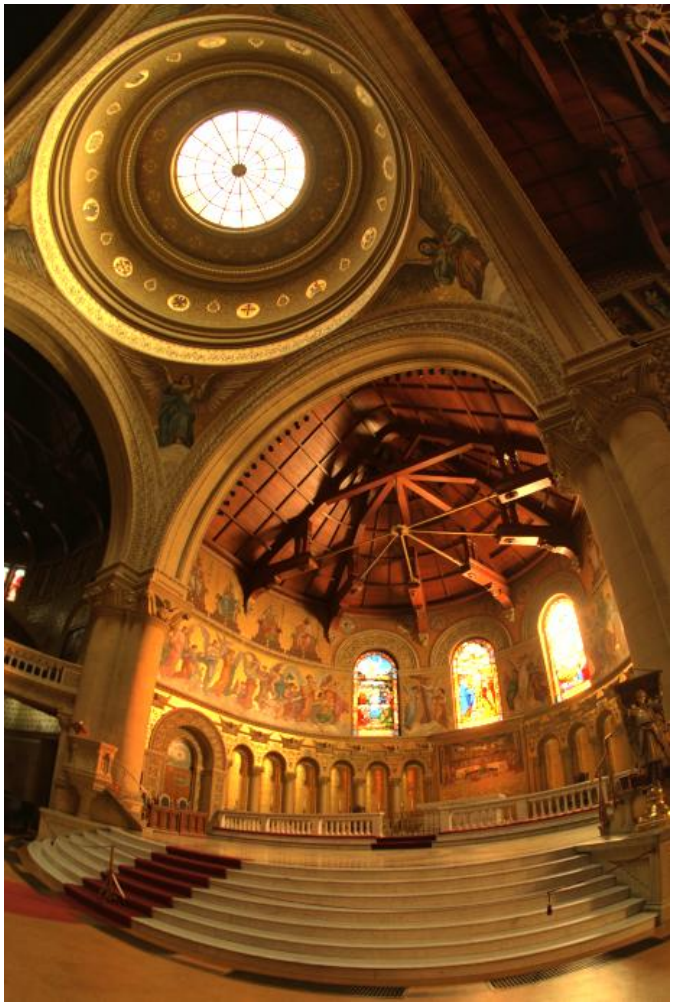

Figure 3: resulting images using openCV

# 2 Task2

In this task, we are required to derive the change in SNR when using the burst denoising method. Specifically, there are two scenarios, each corresponding to different noise distributions

## 2.1 Gaussian-distributed read noise

In this subsection, the only source of noise in a set of K aligned images is zero-mean Gaussian-distributed read noise with a standard deviation of $\sigma$. The previous SNR for each image is $\frac{\mu}{\sigma}$.

The derivation process is as follows.

(1) noise in each image follows Gaussian - distribution

$$\text{noise\_i} \sim N(0, \sigma_i)$$

mean value of each image $\mu_i$

$$\therefore \text{SNR}_i = \frac{\mu_i}{\sigma_i}$$

we summarize these images and average them

$$\text{mean value} = \frac{\sum_{i=1}^{k} \mu_i}{k}$$

$$\frac{1}{k}\sum_{i=1}^{k} N(0, \sigma_i) \sim N\left(0, \frac{1}{k}\sqrt{\sum_{i=1}^{k}\sigma_i^2}\right)$$

$$\therefore \text{SNR} = \frac{\frac{1}{k}\sum_{i=1}^{k}\mu_i}{\frac{1}{k}\sqrt{\sum_{i=1}^{k}\sigma_i^2}} = \frac{\sum_{i=1}^{k}\mu_i}{\sqrt{\sum_{i=1}^{k}\sigma_i^2}}$$

$$\therefore \mu_1 = \mu_2 = \cdots = \mu_k = \mu$$

$$\sigma_1 = \sigma_2 = \cdots = \sigma_k = \sigma$$

$$\text{then SNR} = \frac{k\mu}{\sqrt{k\sigma^2}} = \sqrt{k}\,\frac{\mu}{\sigma}$$

Compare to the previous SNR of each image $(\frac{\mu}{\sigma})$

is $k > 1$, the averaged image has a bigger SNR

meaning a better quality

We can found that after applying burst denoising method, the SNR increased by a factor of $\sqrt{k}$, which enhances the SNR when k > 1 (it is always bigger than 1).

### 2.2 Poisson-distributed photon noise

In this subsection, the only source of noise in a set of K aligned images is Poisson-distributed photon noise. Each pixel observes an average photon rate of $\gamma$, so the SNR of each of the K images is $\frac{\gamma}{\sqrt{\gamma}}$, which equals to $\sqrt{\gamma}$. The derivation process is as follows.

(2) noise_i $\sim P(\lambda_i)$

$SNR\_i = \frac{\lambda}{\sqrt{\lambda}} = \sqrt{\lambda}$

we summarize these images and average them

mean value $= \frac{\sum_{i=1}^{k} \lambda_i}{k} = \frac{1}{k} \sum_{i=1}^{k} \lambda_i$

$\sum_{i=1}^{k} P(\lambda_i) \sim \underbrace{P\left(\sum_{i=1}^{k} \lambda_i\right)}$

standard deviation of noise $= \sqrt{\sum_{i=1}^{k} \lambda_i}$

average these images, the standard deviation of noise $= \frac{1}{k}\sqrt{\sum_{i=1}^{k} \lambda_i}$

$\therefore SNR' = \frac{\frac{1}{k}\sum_{i=1}^{k} \lambda_i}{\frac{1}{k}\sqrt{\sum_{i=1}^{k} \lambda_i}} = \sqrt{\sum_{i=1}^{k} \lambda_i}$

if $\lambda_1 = \lambda_2 = \cdots = \lambda_k$

$SNR' = \sqrt{k} \cdot \frac{\lambda}{\sqrt{\lambda}} = \sqrt{k} \cdot \sqrt{\lambda}$

As just found in (1), the processed SNR is bigger than the initial SNR for each image. indicating a better quality.

Similar to the previous case, after applying the burst denoising methosd, the SNR increased by a factor of $\sqrt{k}$, which enhances the SNR when k > 1.

## 3 Task3

In Task 3, we introduce two different types of cameras, each corresponding to different noise distributions. Combined with two different exposure methods, flutter shutter and burst denoising, there are a total of four scenarios, and we calculate their respective SNRs.

Here is the conclusion table.

| | flutter shutter SNR | Burst SNR |
|---|---|---|
| Consumer camera (n=10000) | 100 | 20 |
| sCMOS camera (n=1000) | $10\sqrt{5}$ | $10\sqrt{10}$ |

The calculation process is as follows.

(1) Consumer camera

① Flutter Shutter

$N = 10000$

$\mu = 0.5 \cdot N = 5000$

$\sigma = 50$

$SNR = 100$

② Burst

for each exposure $N = \frac{10000}{100} = 100$

$\sigma = 50$ $SNR = 2$

take 100 images and average them

$SNR' = \sqrt{100} \cdot SNR = 10 \cdot 2 = 20$

(2) CMOS sensor

① Flutter shutter

$N = 1000$ $\lambda = 0.5 \cdot 1000 = 500$

$SNR = \sqrt{\lambda} = 10\sqrt{5}$

② for each exposure

$N = 1000$ $\lambda = \frac{1000}{100} = 10$

$SNR = \sqrt{10}$.

take 100 images and average them

$SNR' = \sqrt{100} \cdot SNR = 10\sqrt{10}$

From the calculation results above, we can observe that for the first consumer camera with noise following a normal distribution, the SNR obtained by the flutter shutter exposure method is significantly greater than that of the burst method. For the second sCMOS camera with noise following a Poisson distribution, the SNR obtained by the burst exposure method is greater than the SNR obtained by the flutter shutter method.

Vertically, when using the flutter shutter exposure method, the consumer camera yields a higher SNR than the

sCMOS camera. Conversely, when using the burst exposure method, the sCMOS camera obtains a higher SNR than the consumer one.

Therefore, we can conclude that flutter shutter is more effective in handling noise following a normal distribution, while burst is more effective in handling noise following a Poisson distribution.

# A task1-code

```
import numpy as np
import matplotlib.pyplot as plt
import skimage.io as io
import imageio
import cv2
from pdb import set_trace

# initialize HDR image with all zeros
hdr =np.zeros((768, 512, 3), dtype=float)
scale =np.zeros((768, 512, 3), dtype=float)
T = []
with open('hdr_data/memorial.hdr_image_list.txt', 'r') as file:
    # 或者逐行读取
    i =0
    for line in file:
        i +=1
        if i >3:
            T.append(1 /float(line.split(' ')[1]))


##########################################
def gamma(img):
    return img **2.2


def weight(img):
    return np.exp(-4 *(img -0.5) **2 /0.5 **2)


# load LDR images from hdr_dir (need to unzip before)
fig, axes =plt.subplots(nrows=2, ncols=8, figsize=(30, 15))
for i in range(61, 77):
    img =io.imread('hdr_data/memorial00' +str(i) +'.png').astype(float) /255
    weight_i =weight(gamma(img))
    hdr +=weight_i *(np.log(gamma(img) +np.finfo(np.float32).eps) -np.log(T[i -61]))
    scale +=weight_i
    axes[(i -61) //8, (i -61) % 8].imshow(np.clip(weight_i, a_min=0, a_max=1))
    # axes[(i-61)//8, i].set_title(f'blurred image with sigma={}', fontsize=30)
    axes[(i -61) //8, (i -61) % 8].axis('off')
plt.tight_layout()
# plt.savefig("hw4-task1-weights.png")
plt.show()
# fuse LDR images using weights, make sure to store your fused HDR using the name hdr
# hdr = ...
##########################################

# Normalize
hdr =np.exp(hdr /scale)
hdr *=0.8371896 /np.mean(
    hdr) # this makes the mean of the created HDR image match the reference image (totally optional)

# convert to 32 bit floating point format, required for OpenCV
hdr =np.float32(hdr)

# crop boundary - image data here are only captured in some of the exposures, which is why they are indicated in blue
                                                                    in the LDR images
hdr =hdr[29:720, 19:480, :]
s = [0.1, 0.5, 1]
y = [5, 2.2, 1]
hdr0 =hdr
fig1, ax =plt.subplots(nrows=3, ncols=3, figsize=(15, 20))
for i in range(3):
    for j in range(3):
        hdr =(s[i] *hdr0) **(float(1 /y[j]))
        ax[i, j].imshow(np.clip(hdr, a_min=0, a_max=1))
        ax[i, j].set_title(f'scale={s[i]},gamma=1/{y[j]}', fontsize=15)
        ax[i, j].axis('off')
plt.tight_layout()
```

```
plt.savefig("hw4-task1-s_y.png")
plt.show()

# tonemap image and save LDR image using OpenCV's implementation of Drago's tonemapping operator
gamma =1.0
saturation =0.7
bias =0.85
tonemapDrago =cv2.createTonemapDrago(gamma, saturation, bias)
ldrDrago =tonemapDrago.process(hdr0)
io.imsave('my_hdr_image_tonemapped.jpg', np.uint8(np.clip(3 *ldrDrago, 0, 1) *255))

# write HDR image (can compare to hw4_1_memorial_church.hdr reference image in an external viewer)
hdr =cv2.cvtColor(hdr, cv2.COLOR_BGR2RGB)
cv2.imwrite('my_hdr_image.hdr', hdr)
```