Digital Image Analysis – Assignment 1

Part 1

Linear Scaling

Code(in Python)

```
import cv2
import numpy as np
import imageio
def linearScaling(img,outmax,outmin):
    imin = np.min(img)
    inmax = np.max(img)
    img1 = outmin + (img - inmin)*((outmax-outmin)/(inmax - inmin))
    return img1

def gamma(img):
    img1 = (img**(1/2.2))
    img1 = img1*(255/np.max(img1))
    return img1
```

Here outmax and outmin are controllable parameters. In the image below I have linearly rescaled the image between 0 and 255 and saved it in jpeg format. We conclude that when the dynamic range is so huge linear rescaling is not a good option as it leads to losing information about the image. Therefore we use rescaling in the log domain.

After applying linear scaling

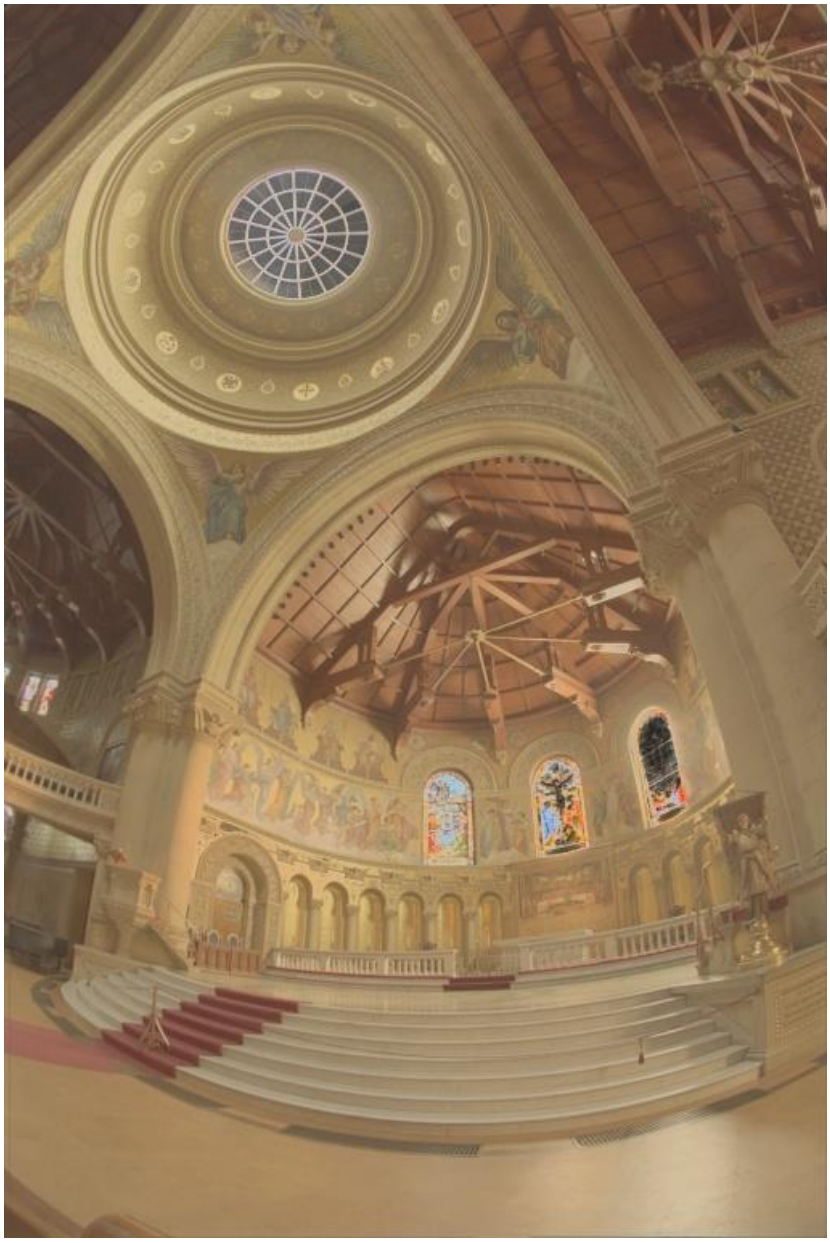
Log Rescaling

Code(in Python)

```
img = imageio.imread('memorial.hdr',format='HDR-FI')

img1 = 0.299*img[:,:,0] + 0.587*img[:,:,1] + 0.114*img[:,:,2]

img5 = img1 + 1

img5 = np.log10(img5)

img3 = img.copy()

img6 = img.copy()
```

```
img8 = img.copy()

img4 = img5.copy()

inmin = np.min(img5)

inmax = np.max(img5)

outmin = 2.3

outmax = 0.3

for i in range(img5.shape[0]):

   for j in range(img5.shape[1]):

      img5[i,j] = outmin + (img4[i,j] - inmin)*((outmax-outmin)/(inmax - inmin))

img5 = 10**(img5)

img6[:,:,0] = ((img3[:,:,0]/(img1+1))**0.5)*img5

img6[:,:,1] = ((img3[:,:,1]/(img1+1))**0.5)*img5

img6[:,:,2] = ((img3[:,:,2]/(img1+1))**0.5)*img5

img7 = img6.copy()

img6 = img6**(1/2.2)

img6 = img6*(255/np.max(img6))

img6 = img6[:,:,::-1]

cv2.imwrite('out1.jpg',img6.astype(np.uint8))
```

The image below has been rescaled in log luminance domain. It is clear that the quality and the features are far more visible in this image rather than the last one. But one of the down sides is that as we have recovered the colour channels we may have lost a bit of information on colour. But this image still appears not to be too sharp. In the next part we will apply techniques like unsharp masking to make the image sharper.



After applying scaling in log luminance domain

Part 2

Unsharp Masking

Here I apply unsharp masking with the help of 3x3 Laplacian Mask whose center is negative and then apply high boost filtering with A = 10 on the enhanced image in the log luminance domain.

Code(Python)

```
def unsharp(img1):
    img = img1.copy()
    k = np.array([[0,1,0],[1,-4,1],[0,1,0]])
```

```
    for i in range(1,img.shape[0]-1):
        for j in range(2,img.shape[1]-2):
            X1B = img1[i-1:i+2,j-1:j+2,0]
            X1G = img1[i-1:i+2,j-1:j+2,1]
            X1R = img1[i-1:i+2,j-1:j+2,2]
            B = np.sum(X1B*k)
            G = np.sum(X1G*k)
            R = np.sum(X1R*k)
            img[i,j,2] = R
            img[i,j,1] = G
            img[i,j,0] = B
    img = img1 - img
    min = np.min(img)
    img = img-min+1
#     print(np.max(img))
    img = img*(255/np.max(img))
    #print(np.max(img))
    return img
```

Part 3

Tone Mapping

In this part I have implemented the Reinhard Tone Mapping Algorithm. This produces far better results than all the images above. But since we have implemented this algorithm in the luminance domain there is some colour distortion due to us recovering the colour channels. Hyperparameters are used as is given in the paper.

Code(in Python)

```
img = imageio.imread('memorial.hdr',format='HDR-FI')
img1 = 0.299*img[:,:,0] + 0.587*img[:,:,1] + 0.114*img[:,:,2]
s = 1.6**6
alpha1 = 0.35
alpha2 = 0.35*0.16
```

```
phi = 8
a = 0.36
img2 = img1.copy()
img3 = img1.copy()
for i in range(img2.shape[0]):
   for j in range(img2.shape[1]):
      img2[i,j] = (1/(np.arctan(1)*((alpha1*s)**2)))*np.exp((-(i**2)-(j**2))/((alpha1*s)**2))
      img3[i,j] = (1/(np.arctan(1)*((alpha2*s)**2)))*np.exp((-(i**2)-(j**2))/((alpha2*s)**2))
h1 = np.fft.fft2(img2)
h2 = np.fft.fft2(img3)
L = np.fft.fft2(img1)
V1 = L*h1
V2 = L*h2
V1 = np.fft.ifft2(V1)
V2 = np.fft.ifft2(V2)
V = (V1-V2)/(((2**phi)*(a*(s**2))) + V1)
L1 = img1/(1+V1)
img4 = img.copy()
img4[:,:,0] = ((img[:,:,0]/(img1))**0.5)*L1
img4[:,:,1] = ((img[:,:,1]/(img1))**0.5)*L1
img4[:,:,2] = ((img[:,:,2]/(img1))**0.5)*L1
img4 = img4**(1/2.2)
img4 = img4*(255)
print(np.max(img4))
img4 = img4[:,:,::-1]
cv2.imwrite('out4.jpg',img4.astype(np.uint8))
```

Image after applying Reinhard tone mapping