High Dynamic Range Imaging
INF01213 Computational Photography – 2020/1

Lucas N. Alegre

April, 2020

1 Task 1 - Retrieving Exposure Times

EXIF (Exchangeable Image File) metadata is data embedded within a photograph file in addition to the actual pixel values, containing information on the settings and conditions in which the photograph was taken. On Windows 10, we can obtain the exposure times presented in the EXIF metadata by clicking with the right mouse button at the file and checking the *Properties* field.

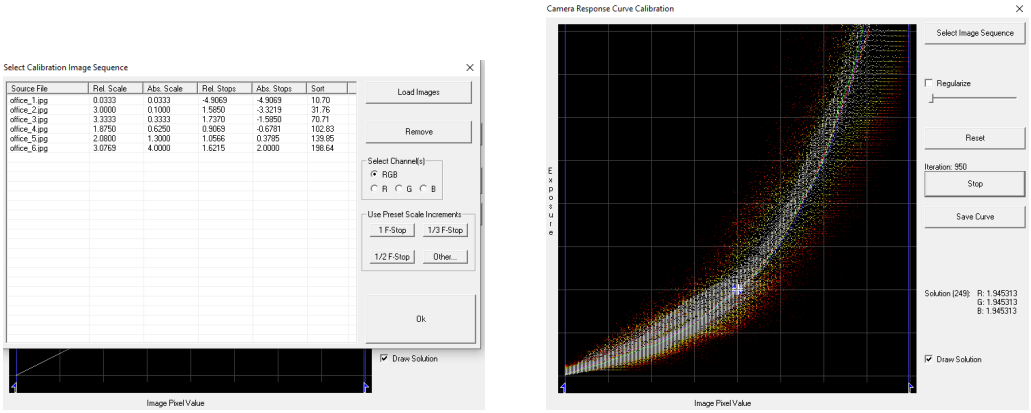
Table 1: Exposure times of each image.

	office_1.png	office_2.png	office_3.png	office_4.png	office_5.png	office_6.png
exposure time (s)	0.0333	0.1	0.3333	0.6250	1.3	4.0

2 Task 2 - Obtain Camera Response Curve

Given the known pixel values and exposure times for each picture in the LDR image sequence, we can plot the camera response curve for a given amount of light, i.e., how it correlates perceived light from the scene to a representation of RGB values.

For this task, I used HDR Shop software, feeding the exposure times in the absolute scale and obtaining the calibration curve, as shown in Figure 1.



(a) Calibration values.

(b) Response curve.

Figure 1: Camera Response Curve.

3 Task 3 - Creating an HDR image from an LDR sequence using HDR Shop

Giving as input to HDR Shop software the exposures times and the camera response curve shown in Figure 1, we can use it to assembly a HDR image. The resulting image is depicted in Figure 2.

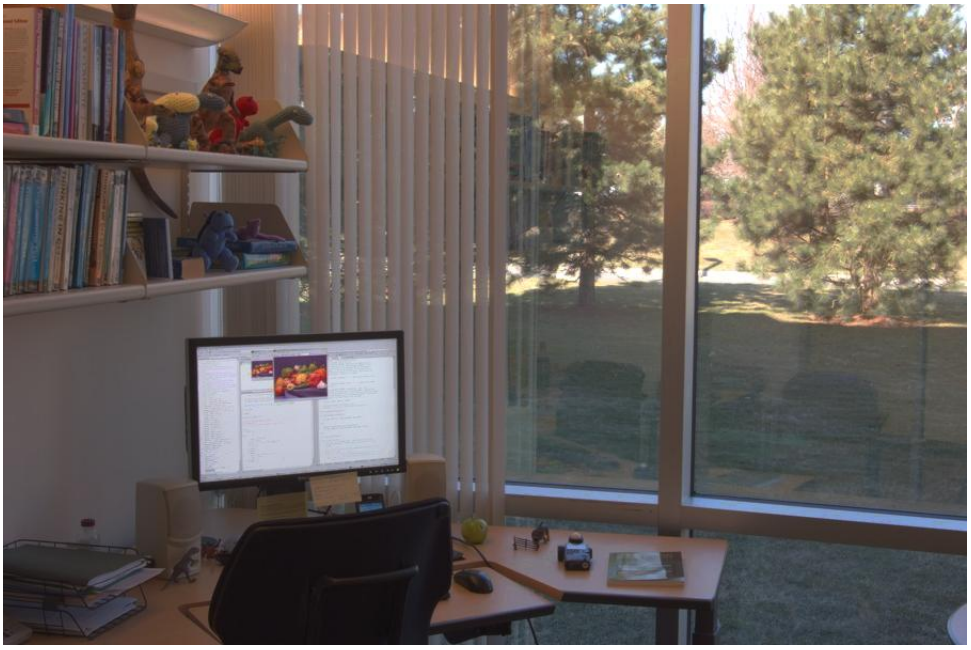


Figure 2: HDR image created with HDR Shop software.

4 Task 4 - Implementing a method to create HDR images

Using the *python* programming language, I implemented the following pipeline:

1. Read all 6 LDR versions of the image with different exposure times and apply gamma decoding with $\gamma = 2.2$.
2. Read the exposure values found in Section 1 and the camera response curve found in Section 2.
3. Compute the irradiances for each LDR image, by retrieving the logarithm of the exposure for each pixel from the response curve, applying exponential and dividing it by the exposure time.
4. Average the resulting irradiances, producing a HDR version of the image. Obs: I did not ignore saturated pixels when computing the average, as I found the results obtained by ignoring saturated pixels to have some strange color distortions.
5. Save the HDR image in .mat format, so that it can be loaded in MATLAB.

This implementation process was very straightforward. The HDR file, after being loaded in MATLAB and saved as PNG is shown in Figure 3.

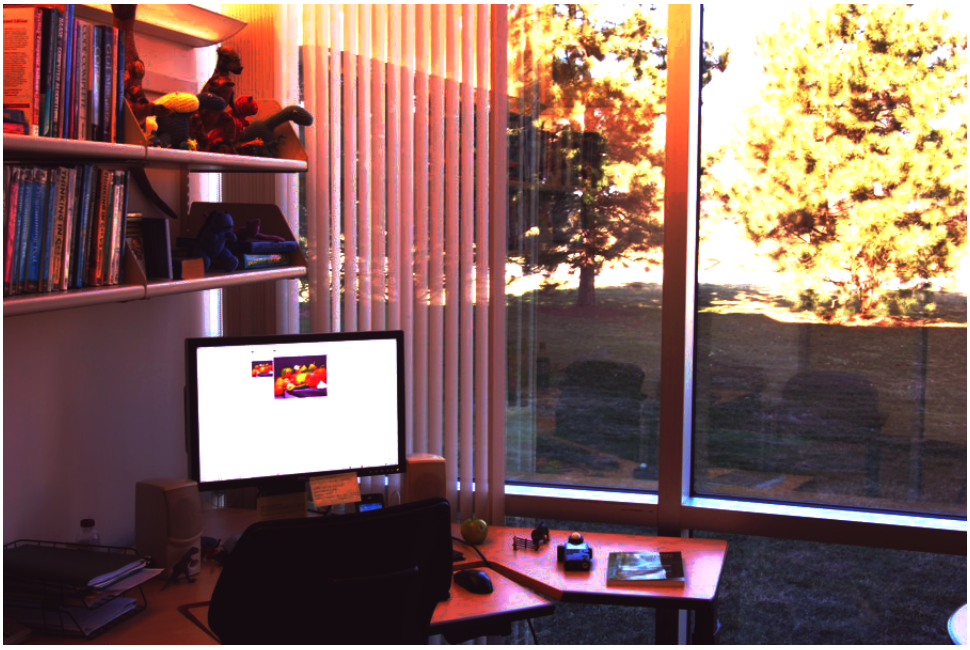


Figure 3: My HDR file (saved as PNG).

I used MATLAB's *tonemap* command to create a tonemapped version of my HDR image. The result is shown in Figure 4.

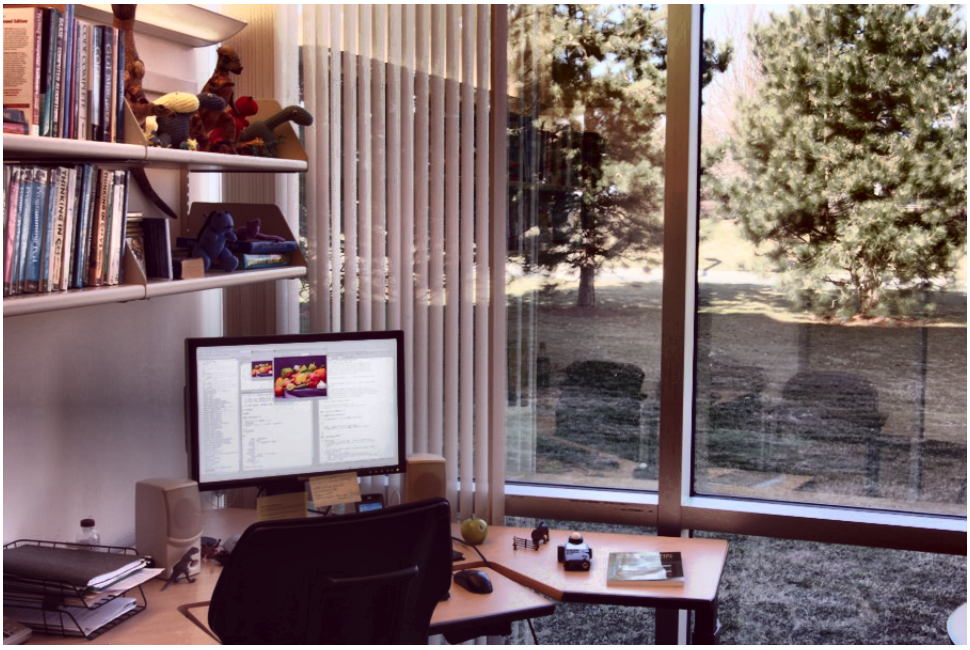


Figure 4: Resulting LDR image of MATLABS's tonemap applied to my HDR image.

5 Task 5 - Using MATLAB's *makehdr* command

Comparing the HDR file created by me (Figure 3) and by MATLAB's *makehdr* command (Figure 5), we can perceive that the resulting image with my implementation resulted in a more saturated image, while MATLAB's version was very bright. It is important to notice, however, that both images were saved as PNG, which can explain the strange results before tonemapping.

After applying MATLAB's *tonemap* command, we can perceive that the HDR file created by my implementation (Figure 4) is more colorful than the one produced by MATLAB (Figure 6), which got a very white-grey aspect.

Figure 5: HDR image created with makehdr command (saved as PNG).

Figure 6: LDR tonemapped version of HDR image created by MATLAB.

6 Task 6 - Implementing Reinhard's global tone mapping operator

For this part of the assignment, I implemented the Reinhard's Global Tone Mapping operator.

The algorithm averages out the log luminance of each pixel ($\widetilde{L}$), which will be used as a uniform scaling factor for our luminance values:

$$L_s = \frac{\alpha}{\widetilde{L}} L(x, y). \tag{1}$$

where $L(x, y) = 0.299R(x, y) + 0.587G(x, y) + 0.114B(x, y)$ and $\alpha = 0.18$, as suggested in Reinhard's paper *Photographic Tone Reproduction for Digital Images*.

The Reinhard's global operator is then:

$$L_G = \frac{L_s}{1 + L_s}. \tag{2}$$

The final, still linearized, RGB values of the tonemapped image is then obtained following:

$$RGB_L = (L_G(R/L)^s, L_G(G/L)^s, L_G(B/L)^s) \tag{3}$$

where the exponent s controls the saturation of the colors and is typically in the range $s \in [0.4, 0.6]$ (see Chapter 10 of Richard Szeliski's book *Computer Vision: Algorithms and Applications*). Finally, we apply gamma encoding to obtain the final image.

In the following figures, I show the tonemapped images obtained by my implementation with different values for the saturation s. With $s = 0.6$ (Figure 9, I found that the resulting image is very similar to the HDR image created by HDR Shop software (Figure 2). In particular, I prefer my results than the ones produced with MATLAB. I suspect that MATLAB results may have been prejudiced for not having as input the exposure times and/or the response curve calculated with HDR Shop software.

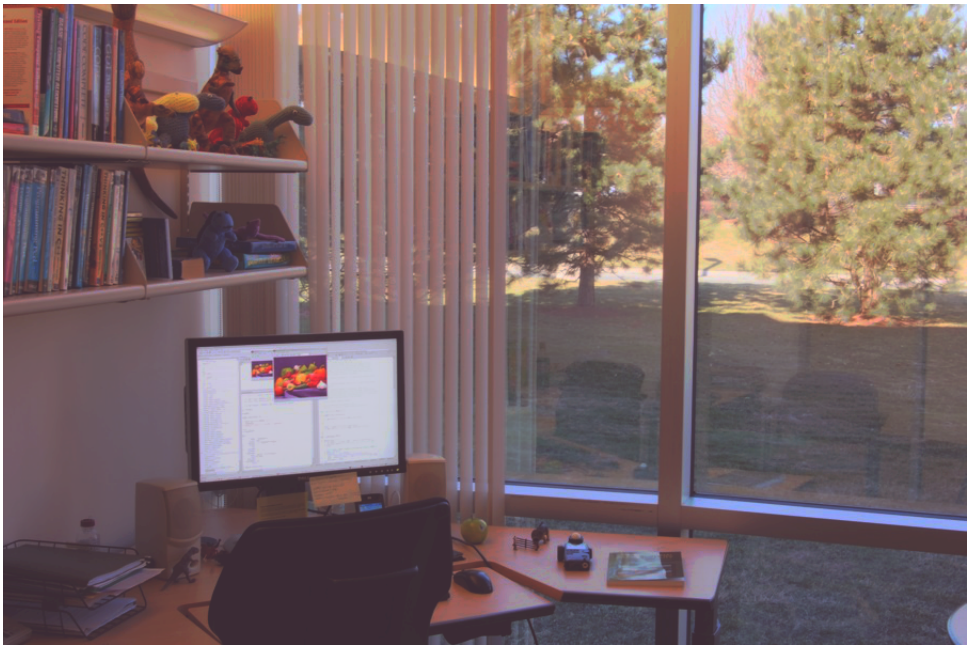


Figure 7: Tonemapped HDR image ($s = 1$).

Figure 8: Tonemapped HDR image ($s = 0.8$).

Figure 9: Tonemapped HDR image ($s = 0.6$).

Figure 10: Tonemapped HDR image ($s = 0.4$).

7 Conclusions and considerations

All tasks of the assignment were successfully implemented. The implementations were very straightforward, and the biggest difficulties were found in the process of understanding why Reinhard's tonemapping algorithm is the way it is. In particular, Equation 3 was brought by my as topic of interesting discussions in class. The saturation exponent trick found in Szeliski's book helped me to improve my results, since that without it (or equivalently, $s = 1$) the resulting image is very reddish (Figure 7).

I decided to follow Szeliski's book and avoid the process of converting from RGB to XYZ and Yxy color spaces motivated by an interesting discussion that I found on internet: `https://imdoingitwrong.wordpress.com/2010/08/19/why-reinhard-desaturates-my-blacks-3/#comments`.

This assignment helped me to better understand how cameras translate the perceive light as RGB values, and how can we construct HDR images from a set of LDR images with different exposure times. The final results were very good and not too far from the HDR Shop software implementation, in my opinion.