Name: Batuhan

Surname: Yalçın

ID: 64274

COMP527/ELEC519 – Programming Assignment 2: HDR Imaging and Tone mapping

1 HDR imaging)

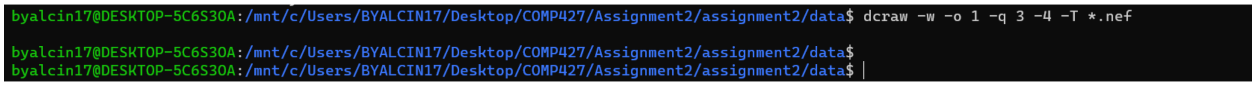
Develop RAW image:

From the dcraw guideline flags declaration found to convert in described way:

• w : Use the camera's white balance settings
• o 1 : to output since it is the sRGB output space
• q 3 : To interpolation during demosaicing using adaptive method.
• 4 : To linearized 16 bit output
• T : To save the output in TIFF format

Then I run following code to convert all nef images:

dcraw −w −o 1 −q 3 −4 −T ∗ . n e f



```
byalcin17@DESKTOP-5C6S3OA:/mnt/c/Users/BYALCIN17/Desktop/COMP427/Assignment2/assignment2/data$ dcraw -w -o 1 -q 3 -4 -T *.nef

byalcin17@DESKTOP-5C6S3OA:/mnt/c/Users/BYALCIN17/Desktop/COMP427/Assignment2/assignment2/data$
byalcin17@DESKTOP-5C6S3OA:/mnt/c/Users/BYALCIN17/Desktop/COMP427/Assignment2/assignment2/data$ |
```

Linearize rendered images:

In this section, I aim to recover the function g(·) to linearize the rendered images for HDR generation in later parts for nonlinear inputs such as jpg files. It is crucial to obtain the correct function, which depends on the weighting scheme and the smoothness factor. After several try, I choose smoothness factor lambda (λ) as 10. All the outputs mainly in between -6 and 6. However, when using the tent weighting function, the function assigns higher weights to pixels with good exposure, resulting in skewed output values of g towards higher values. This is further influenced by the regularization term in the optimization problem. Although some points have extremely high tent weights, the output values of g mainly fall between -6 and 6. The plot of the g functions are in below:

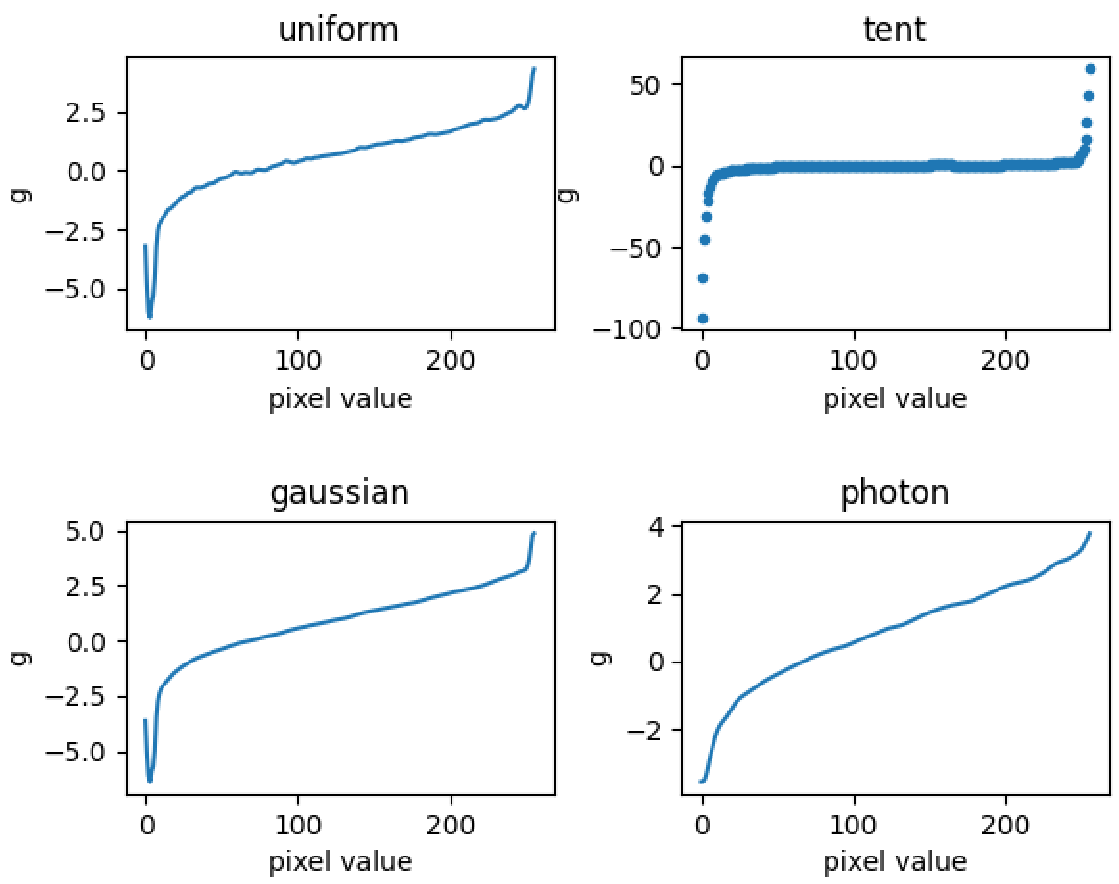


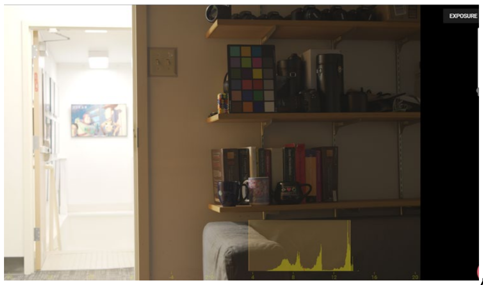
Figure 1 Linearization plot of g functions. All of them between mainly -6 and 6. Some small tent function values are bigger than this range see above for explanation.

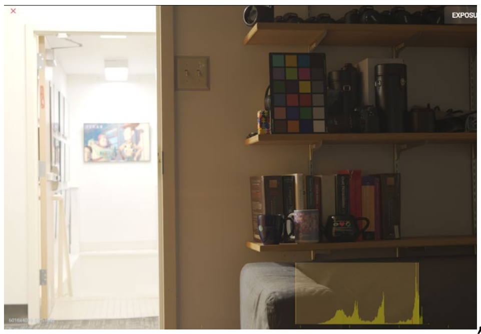
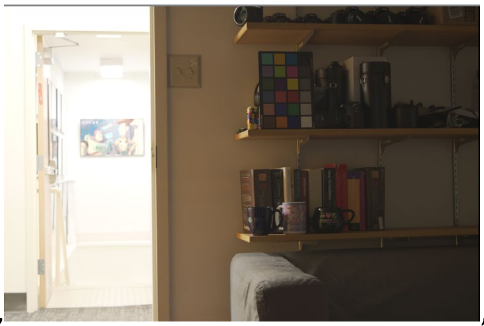
Merge exposure stack into HDR image:

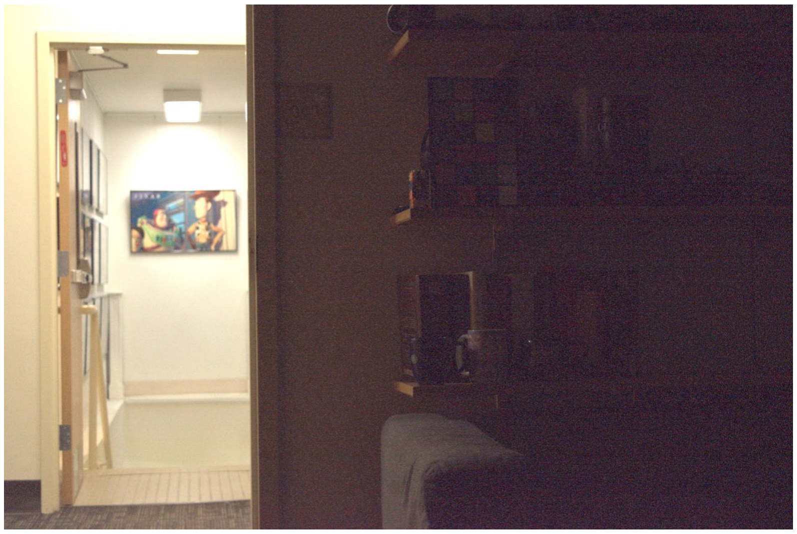
In this part to merge exposure stack into HDR image I use various type of combination. 2 types of pictures. 4 type of weight scheme and 2 type of merging as linear and logarithmic. Total 16 combination gathered. To distinguish linear and logarithmic merging, I use equation 5 and 6. Two distinguish pictures I use linearization for jpg not for tiff I also normalize the jpg with 255 and tiff with 2^16-1 (65535) since jpg pictures are 8 bit and tiff are 16 bits. To weight schemes Gaussian, Tent, Uniform and Photon schemes applied.

Weighting schemes:

For the weighting schemes using different kinds of the clip range for Zmin and Zmax were important to eliminate bad pixels. Since the varying z values for all 16 image was creating too much combination I choose the best image for 0.05 and 0.95 then I varied the values in this phot then try to get optimal value.

, ,

, ,



In given figures I try to change z range from 0.001-0.999 to close to 1 each time I increase the range I got more bad pixels (you can see in last image) after below 0.1-0.9 were better below 0.01 and 0.99 I couldn't see the effect these can be related the my masking operations. I got also satisfied with the Zmin =0.05 and 0.95 and not see signifcan effect of change with close values to these values. So this values are used for the all weight scheme.

I also mask the vbad points as described in the hints:

"When merging many LDR images to HDR ones, you may end up with pixels for which there

are not any well-exposed values (i.e., the sum of weights in the denominators of Equations

(5)-(6) is exactly 0). You can set those pixels to equal the maximum or minimum valid pixel

value of your HDR image, respectively for problematic pixels that are always over-exposed or always under-exposed" I separate the image from the middle pizel of 128. Pixels then with lower this and denum == 0 I change their pixels to to equal the maximum or minimum valid pixel value of my HDR image.

Weighting schemes:

After merging pictures, the images are opened with the https://viewer.openhdr.org/

The main elimination criteria for me to gathering best luminance in color picker since we will use it in the color correction part. The linear images were better for color picker details. Best west schemes were uniform and photon. Then I chose photon weight scheme tiff images and linear merging.

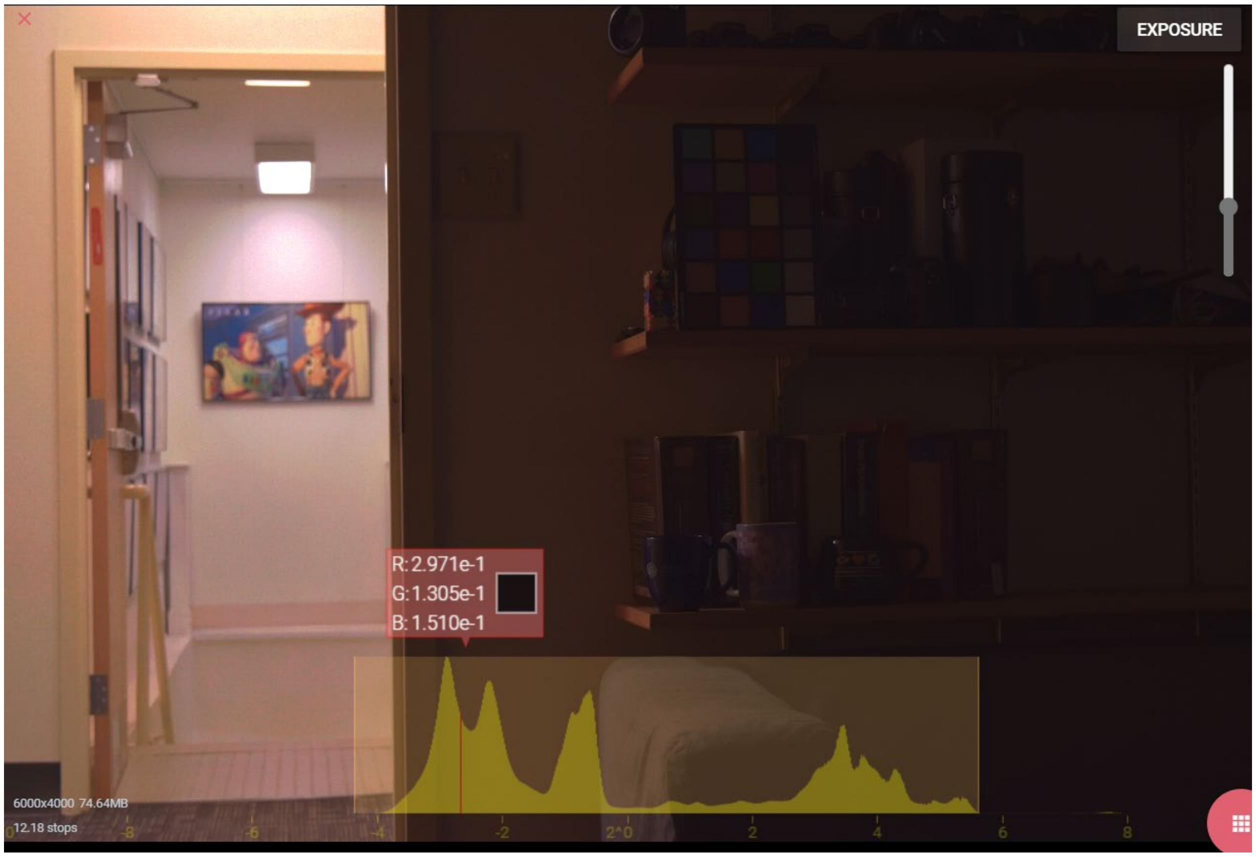


Figure 2 uniform weight scheme jpg images and logarithmic merging

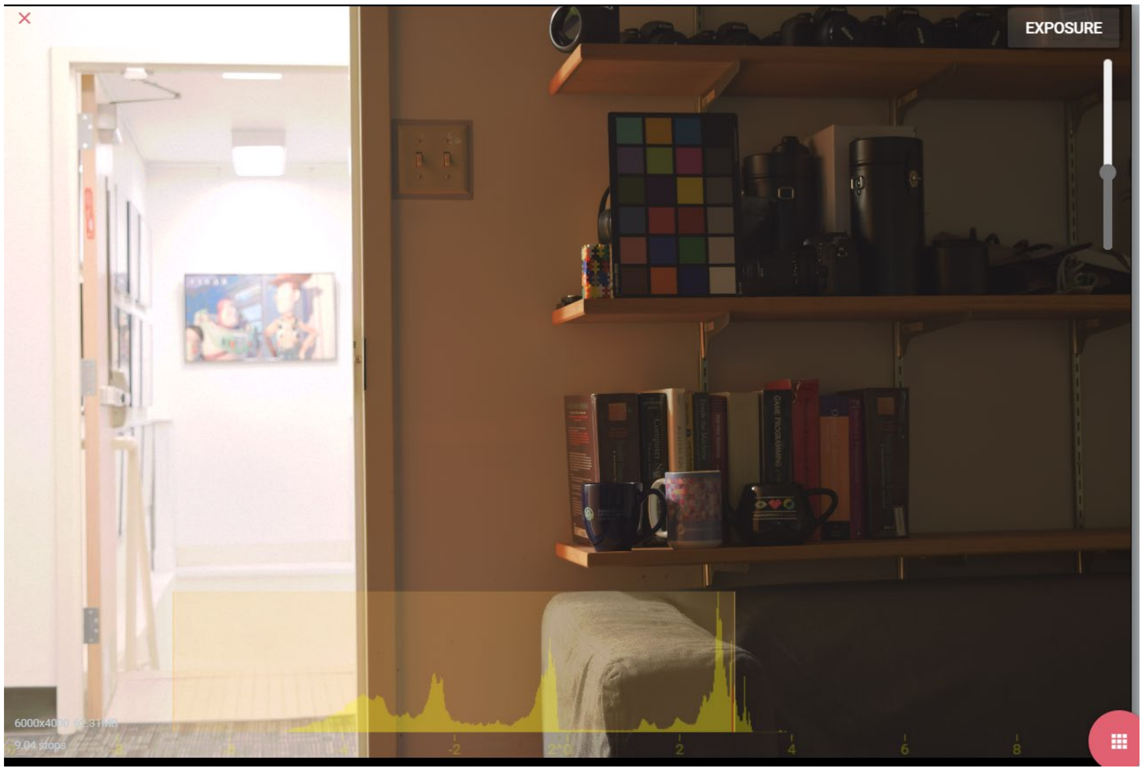


Figure 3 Uniform weight scheme tiff images and linear merging

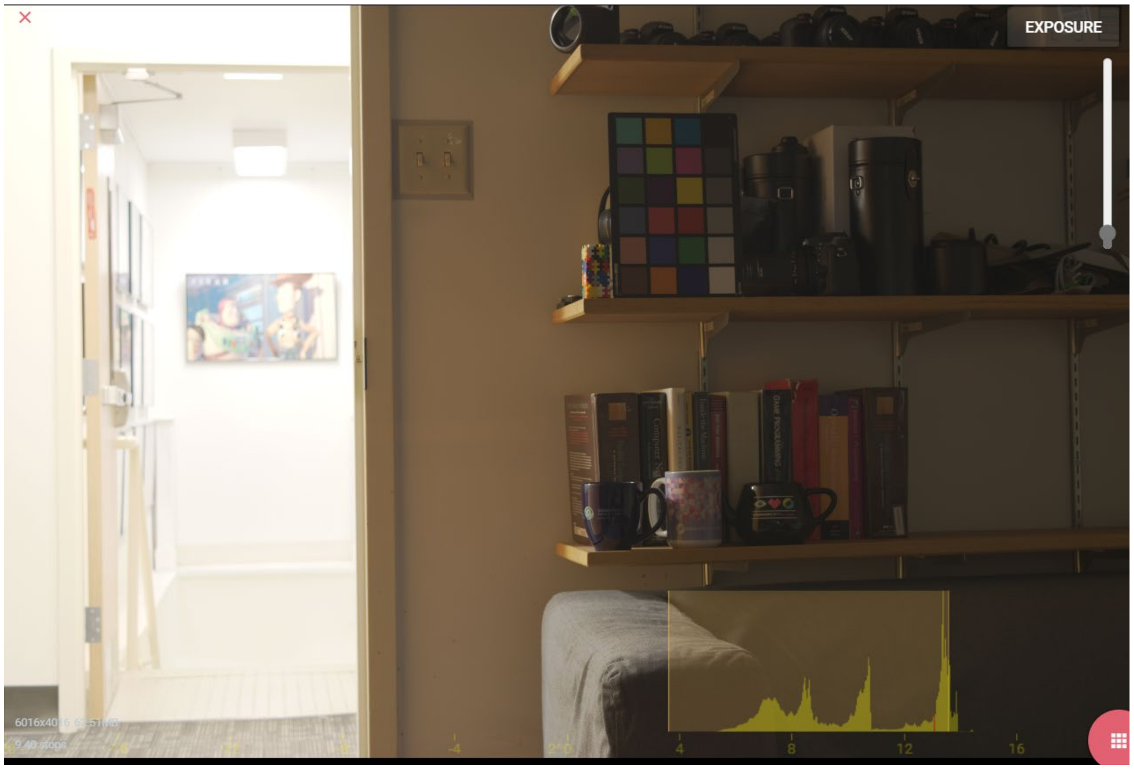


Figure 4 Chosen one photon weight scheme tiff images and linear merging.

2 Color Correction and white balancing)

For this part, I will go over the steps as described in the PDF.

First, I use the matplotlib.pyplot.ginput function to select the color palette colors in the image, column-wise from left bottom to right top. Then, I crop the square with a patch size of 8 to get their average.

Second, I append the bottom truth and patched values with ones to get homogeneous 4 × 1 coordinates.

Third, I solve the least square problem using the np.linalg.lstsq(A_matrix, b_vector) function.

Fourth, I apply transformation by matrix multiplication with the original image.

Finally, I apply the white balance algorithm by getting the ratio with the reference D65 value and current RGB values. By multiplying this ratio factor with all channels of our image, I apply the white balance algorithm.

Here is the result:

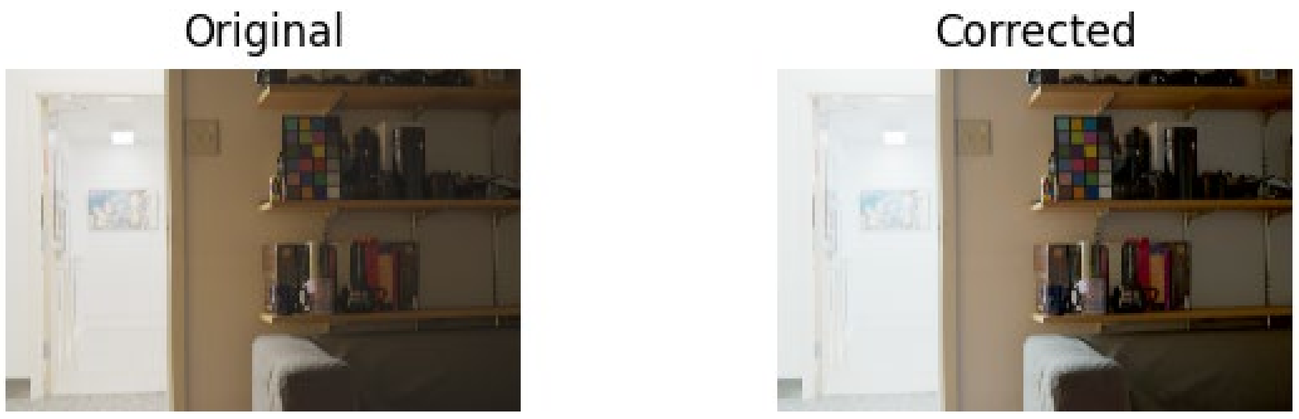
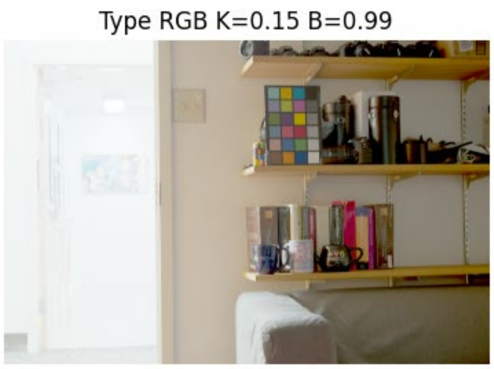
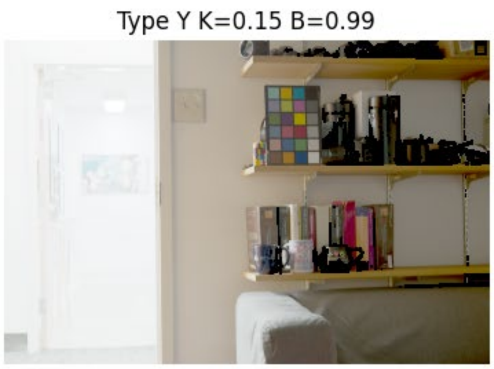


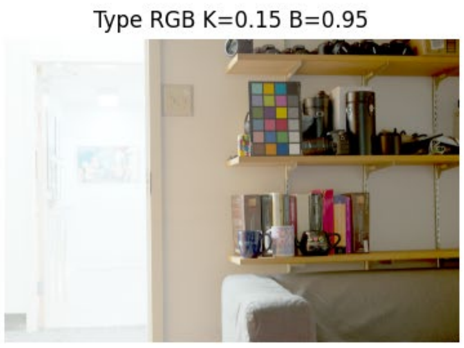
Figure 5: Tonemapped HDR image without (left) and with (right) color correction.

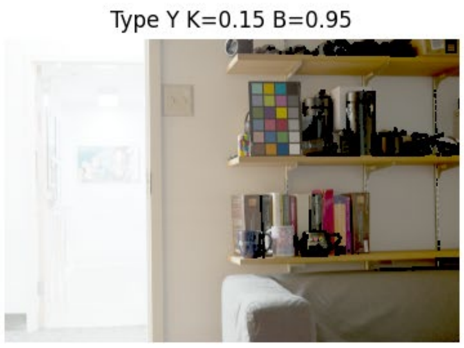
3 Photographic tonemapping)

After experimenting with various combinations of K and B parameters and applying them to all channels or only the luminance channel Y, I made several observations. Firstly, I noticed that B has a direct effect on the contrast of the output image. When B is set above 1, the colors become washed out, while setting it below 0.75 results in overly sharp colors, with the light areas becoming too light and dark areas becoming too dark. After testing different values, I found that setting B to 0.99 produced the best results. Secondly, I found that the K parameter is related to the brightness of the output image. Increasing K made the image much lighter, so I chose a small value of K (K=0.03) for my purposes. It's worth noting that the tone-mapping process utilizes a specific color space to achieve its desired effect, aiming to replicate the color temperature of the input image. However, when performed in the RGB domain, the resulting tone-mapped image may exhibit a color temperature that differs from the original input. Despite this, I opted for the RGB domain over the luminance Y channel.

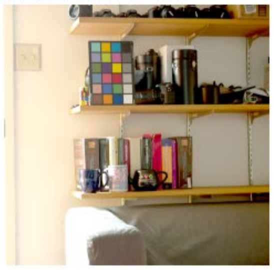




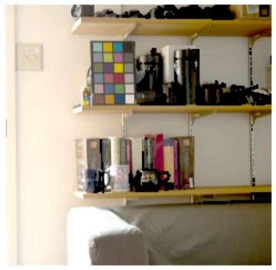




Type RGB K=0.15 B=0.6



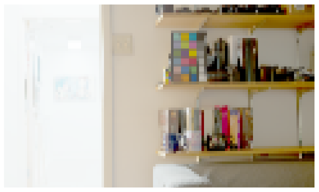
Type Y K=0.15 B=0.6



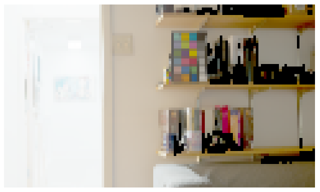
Type RGB K=0.15 B=0.3

Type Y K=0.15 B=0.3

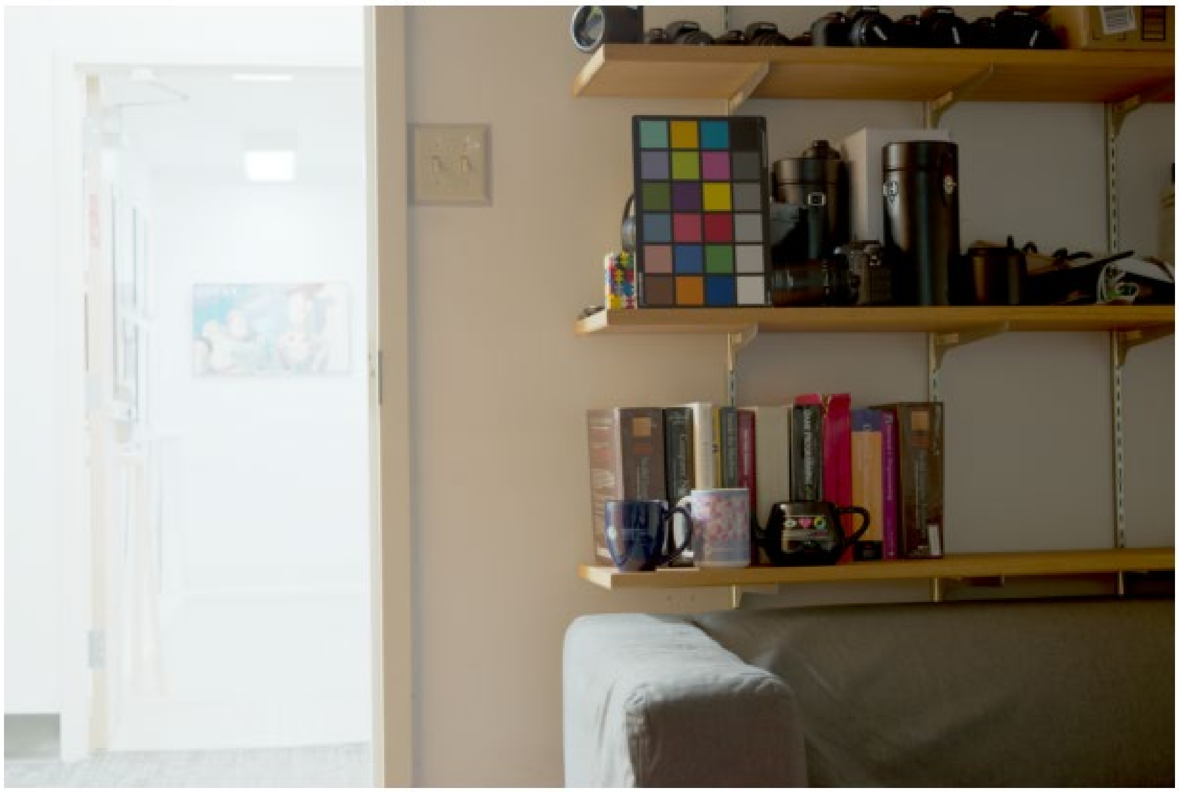
Type RGB K=0.30 B=0.99



Type Y K=0.30 B=0.99



Type RGB K=0.03 B=0.99



The final selection was K=0.03, B=0.99 and RGB channel. But I lose some details in light side, but I focus to color picker, and carpet. Since it can subjective this image was satisfying for me.