Homework Assignment 2

Instructions to Run

```
python hdr.py --args

Args:
--scr_dir: Path to exposure stack directory (Required)
--process: Include if you want to process the raw files
--merge-all: Include to get all merged combinations
--colorcorrect: Include to get all color corrected combinations. Assumes
file you are color correcting is named "before_cc.hdr" and that there is a
numpy file named "crops.npy" both in the same directory. Writes color
corrected hdr to "./out_hdrs/after_cc.hdr".
--tonemap: Include to tonemap hdr saved to "./out_hdrs/after_cc.hdr"
--capture_dark_frames: Include to capture dark frame images. Change
gphoto2 settings to fit scene
--capture_ramp_frames: Include to capture ramp images. Change gphoto2
settings to fit scene.
--get_dark_frame: Include to perform dark frame computation.
--noise_calibrate: Include to perform noise calibration.
```

1. HDR imaging

Develop RAW images

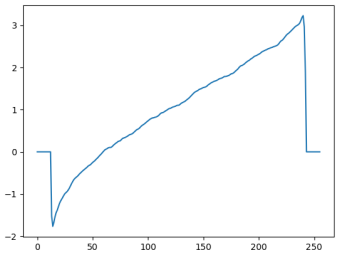
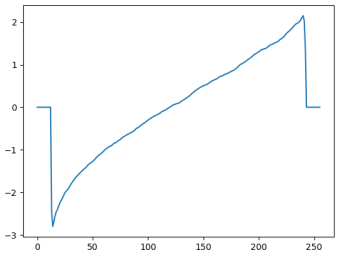
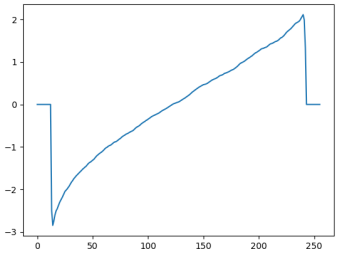
Implemented in `process_raw()` in `hdr.py`.

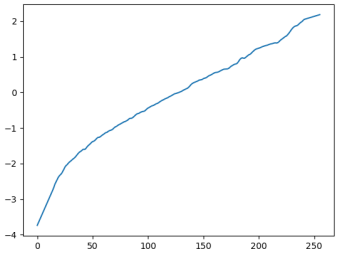
```
dcraw -w -o 1 -q 3 -T -4 -H 0 *.nef

-w: Use the white balance specified by the camera.
-o 1: Color correct to the sRGB colorspace.
-q 3: Use Adaptive Homogeneity-Directed (AHD) interpolation for
demosaicing.
-T: Write TIFF with metadata instead of PGM/PPM/PAM.
-4: Linear 16-bit.
-H 0: Clip all highlights to solid white.
```

Linearize rendered images

Implemented in `linearize()` in `hdr.py`. The recovered ***g*** for each weighting scheme is shown below along with their regularization weights.

Weighting Scheme	Regularization Weight	Recovered g
Uniform	4.0	
Tent	4.0	
Gaussian	4.0	

Weighting Scheme	Regularization Weight	Recovered g
Photon	20.0	

Selected HDR Image

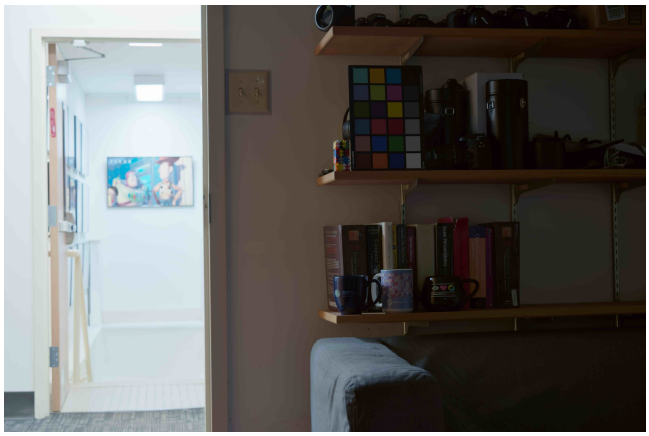
The HDR image I selected had the following configuration:

Image Type	Merging Scheme	Weighting Scheme
jpg	log	gaussian

I selected this HDR image because it had the least amount of clipping in underexposed regions and least amount of color artifacts, and was also the most aesthetically pleasing to me.

2. Color correction and white balancing

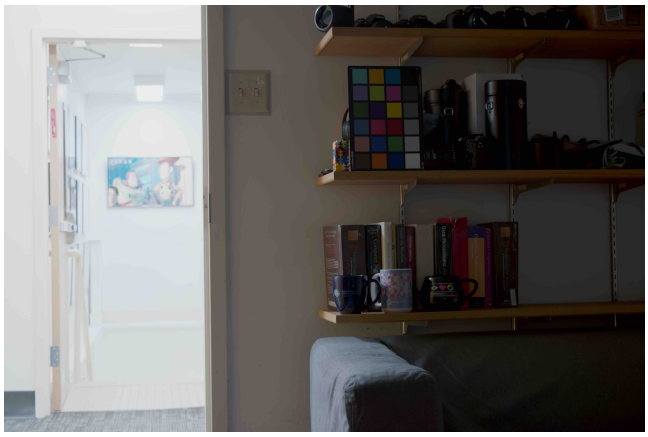
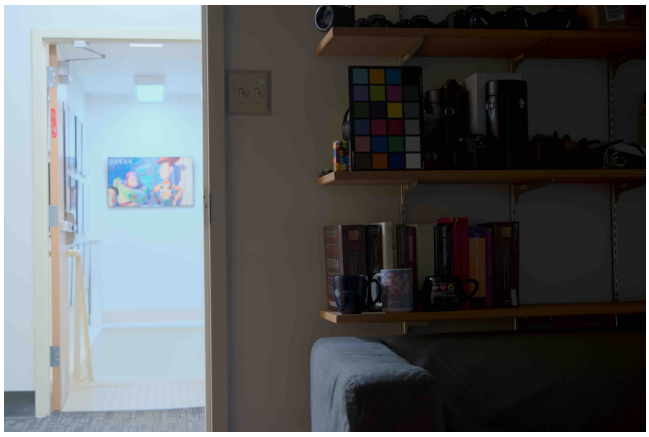
Below are the tonemapped HDR images before and after color correction

Before	After
	

I actually prefer the image before color correction, because it seems to be more in line with how I would actually perceive it if I was there. The color corrected image makes the light in the exterior of the room too white in my opinion. Although that is likely more correct, this is just a matter of personal preference.

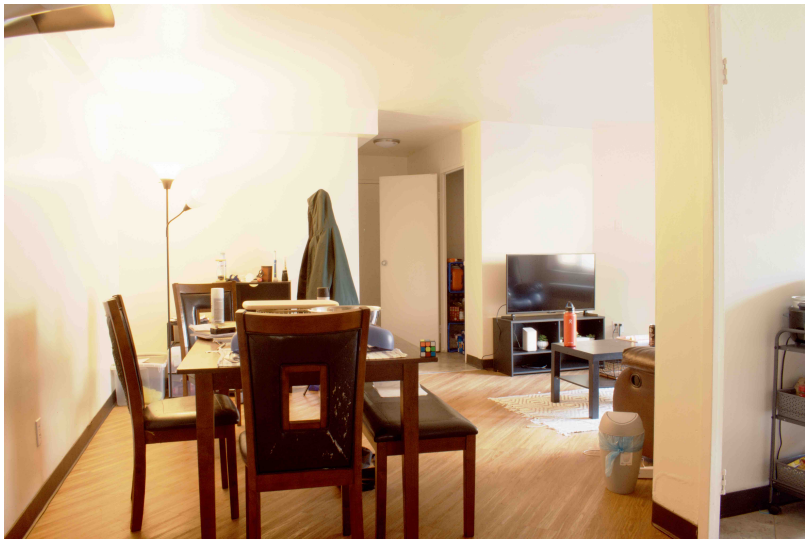
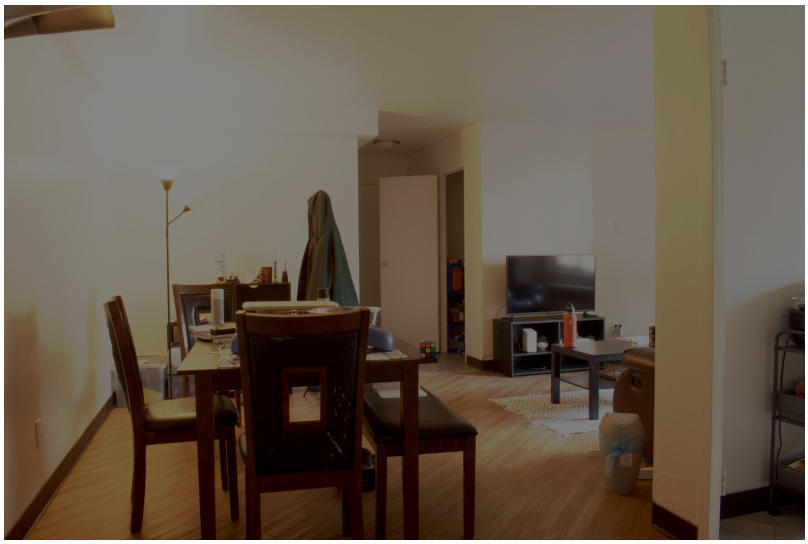
3. Photographic tonemapping

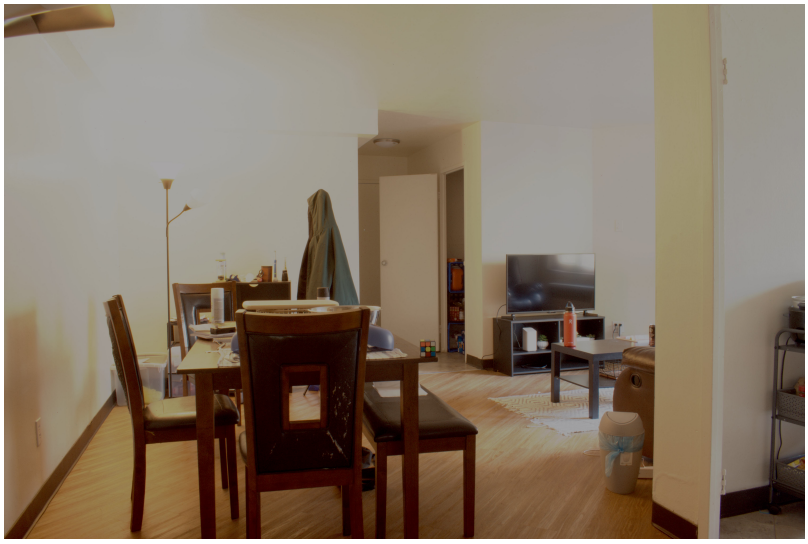
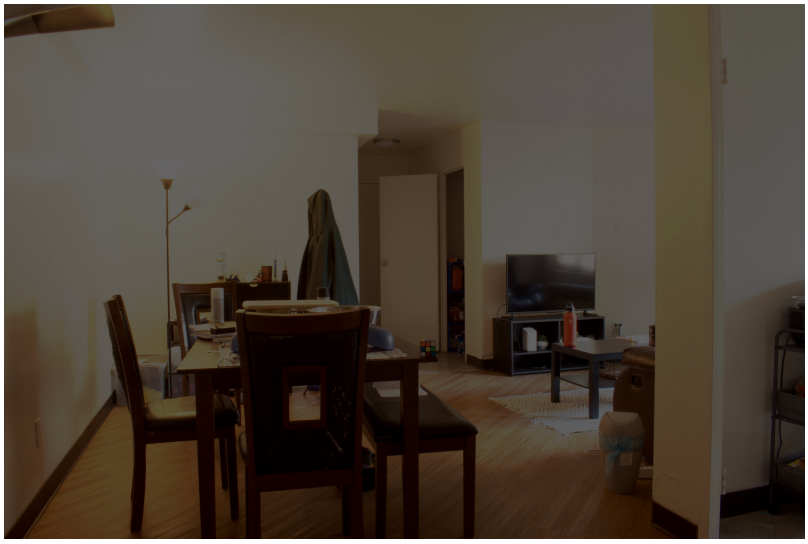
Below are representative tonemaps with K = 0.05 and B = 1.25.

RGB	Luminance
	

4. Create and tonemap your own HDR photo

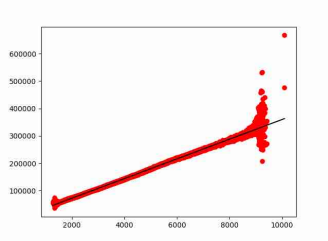
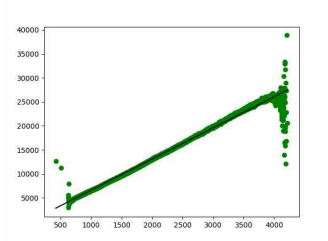
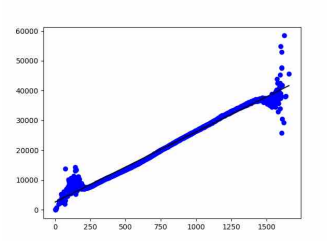
Below are some tonemaps for my own captured exposure stack along with their parameters. The tonemap I liked best had parameters K = 0.01 and B = 1.0 as it didn't mute the image and captured the black colors in the scene very well.

K	B	Tonemap
0.01	1	
0.05	4	

K	B	Tonemap
0.125	6	
0.025	7	

5. Noise calibration and optimal weights

Below are the mean and variance plots for each channel, along with their fitted line and estimated values for gain and additive noise.

	Red	Green	Blue
Gain	35.91713454375651	6.352994321756792	23.52849247392018
Additive Noise	518.5012691093725	142.0744964823043	2649.807828383473
Mean-Variance Plot			

Below is a comparison between the best tonemap I got without optimal weights, and the best tonemap I got with optimal weights. The tonemap with optimal weights seems to really blow out the areas with high intensity ie. the lamp in the back left corner. However, it also seems to capture the black color on the chair more accurately than the tonemap using gaussian weights and logarithmic merging. The parameters for these tonemaps were determined with a grid search over different K and B values.

Gaussian, JPG, Log	**Optimal Weights, Raw, Linear**