COMP 790.175 Computational Imaging and Displays
Assignment 3: ISP

Emre Onemli

1 Developing RAW images

1.1 Implement a basic image processing pipeline

RAW image conversion

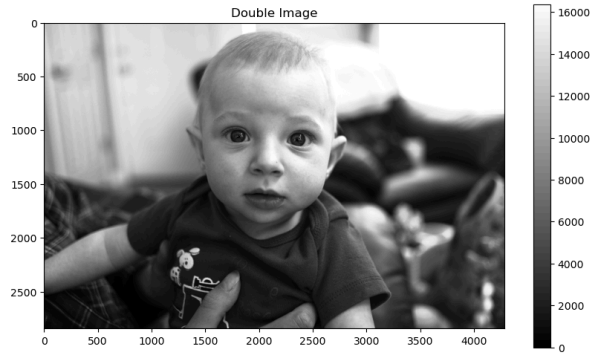
Information about the RAW image after dcraw call:

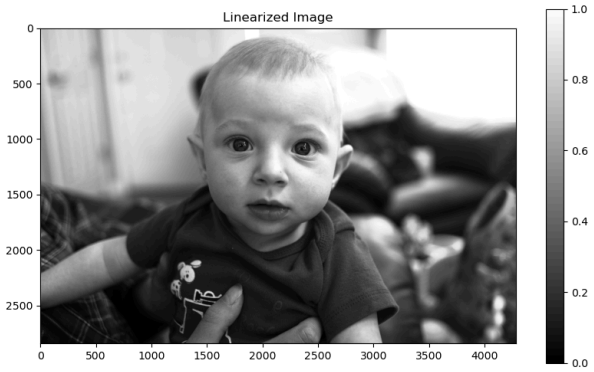
```
Loading Nikon D3 image from ../data/baby.nef ...
Scaling with darkness 0, saturation 16383, and
multipliers 1.628906 1.000000 1.386719 1.000000
Building histograms...
Writing data to ../data/baby.tiff ...
```

Python initials

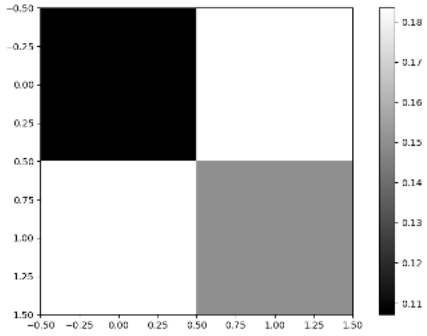
```
Width: 4284, Height: 2844, Bits per pixel: 16
New dtype: float64
```

Linearization





Identifying the correct Bayer pattern



Top-left 2x2 pixels:

[[0.10712324 0.1833608]
[0.18342184 0.14978942]]

Right top and left bottom squares are at a similar brightness, so I think those are green.

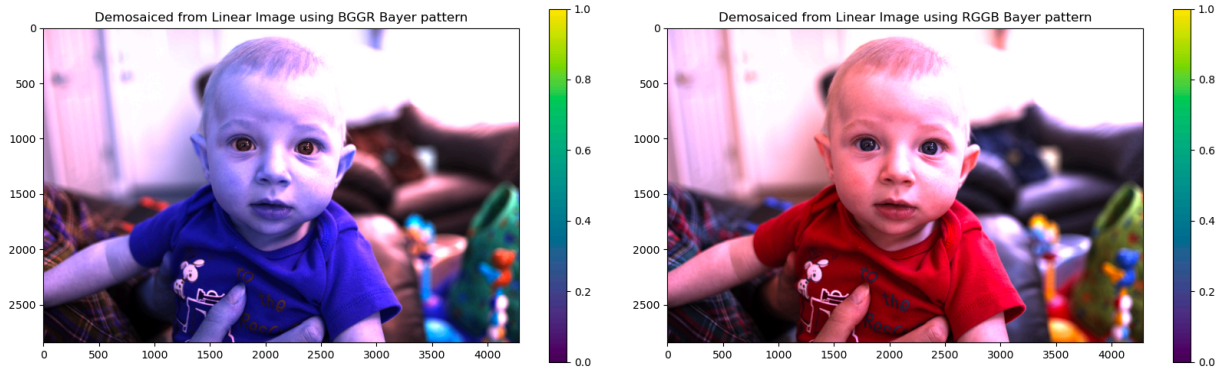
The highest value is 0.18342184, which could indicate a green pixel due to 1. green's higher perceived brightness in most lighting conditions, 2. two pixels are similar

The closest value, 0.14978942, could correspond to the blue pixel

The remaining value, 0.10712324, would then be the red pixel.

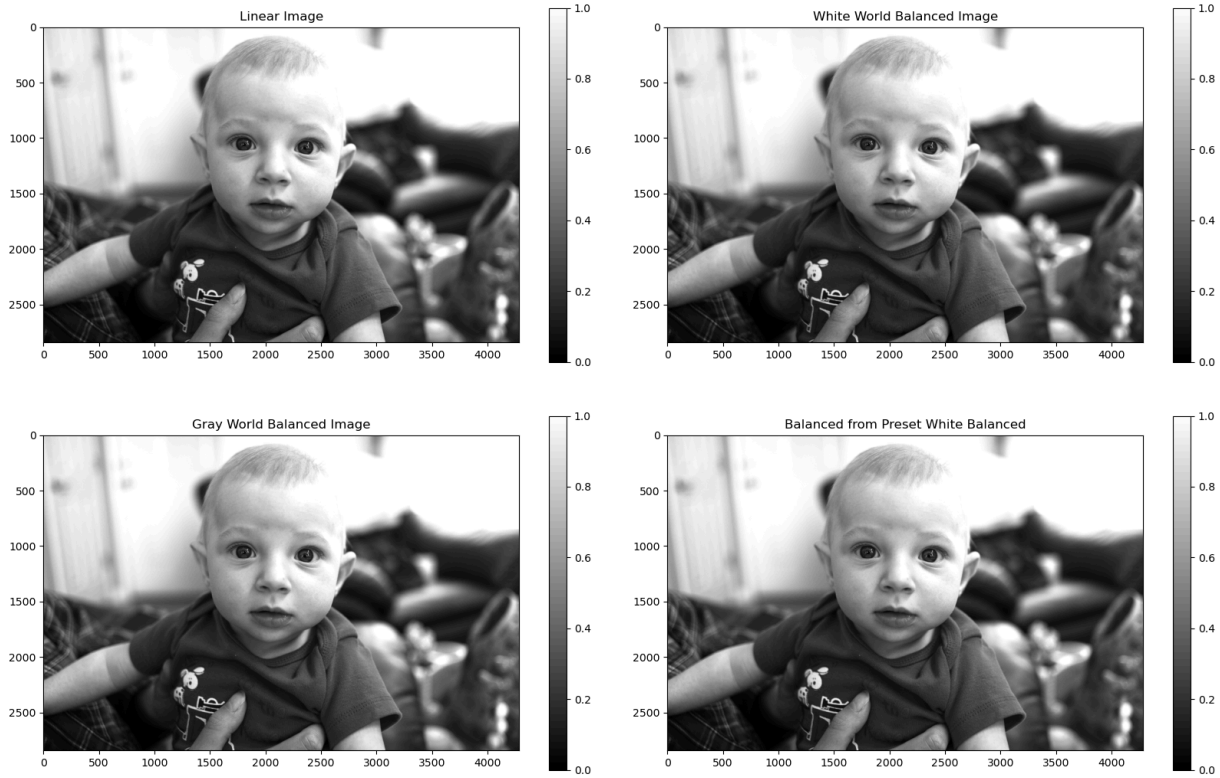
The third pattern **"rggb"** is the most probable Bayern pattern in the image.

The following figure shows the demosaiced images using the “bggr” and “rggb” Bayer patterns.

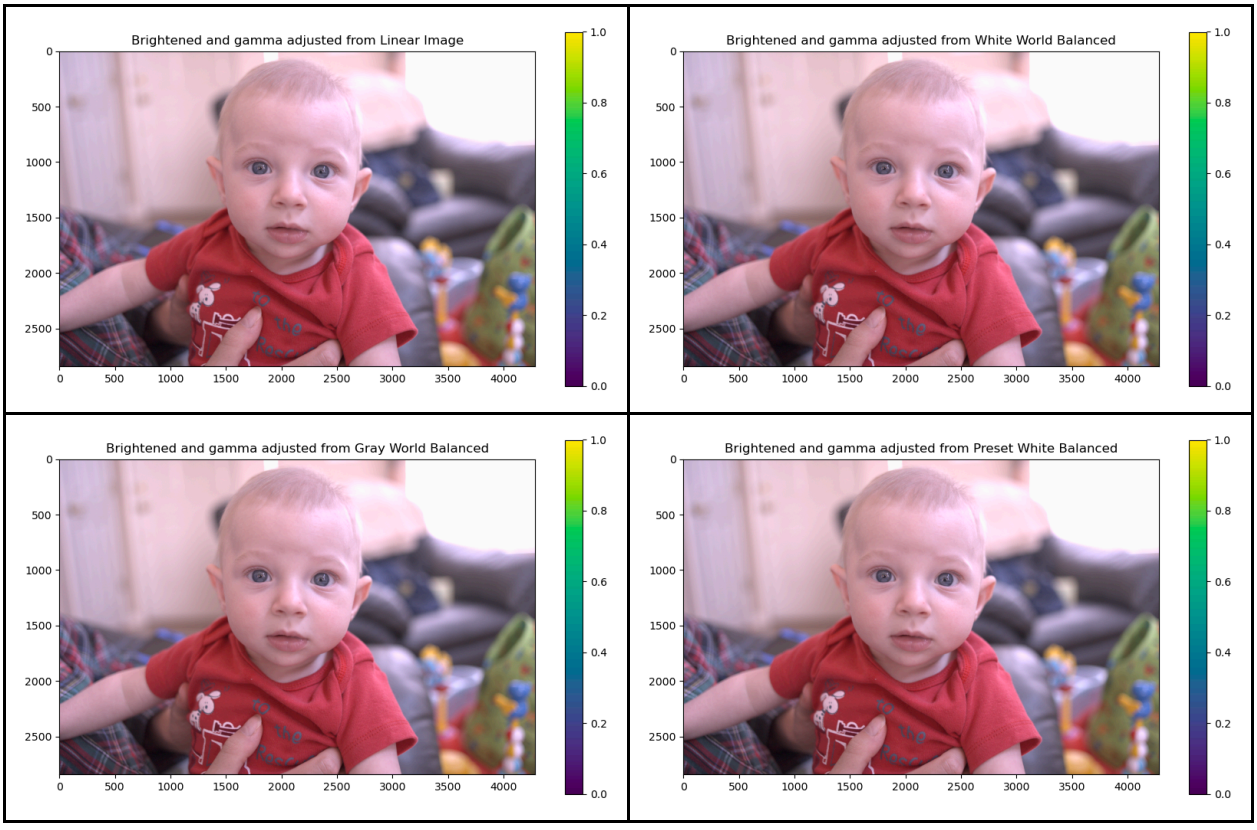


White balancing

The following figure shows the linearized image, after implementing both the white world and gray world and scaling with recorded preset values white balancing algorithms.



The following figure shows the final image results using the same algorithms.

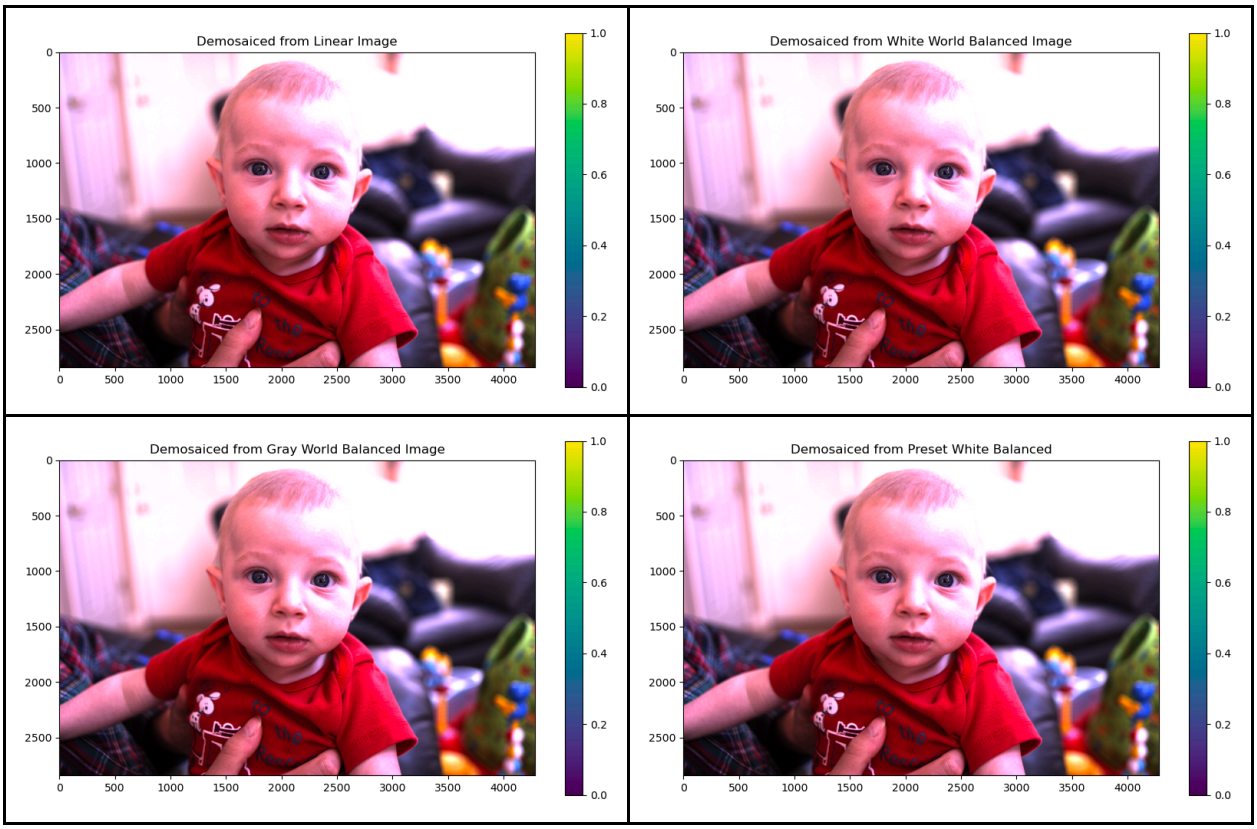


Although the results are very similar, it appears that the **white world algorithm** has done a good job of removing the color cast, producing a more neutral and natural-looking image. It handled bright regions by assuming they should be white or neutral. For example the border of the couch and baby's head is more distinguishable compared to scaled values.

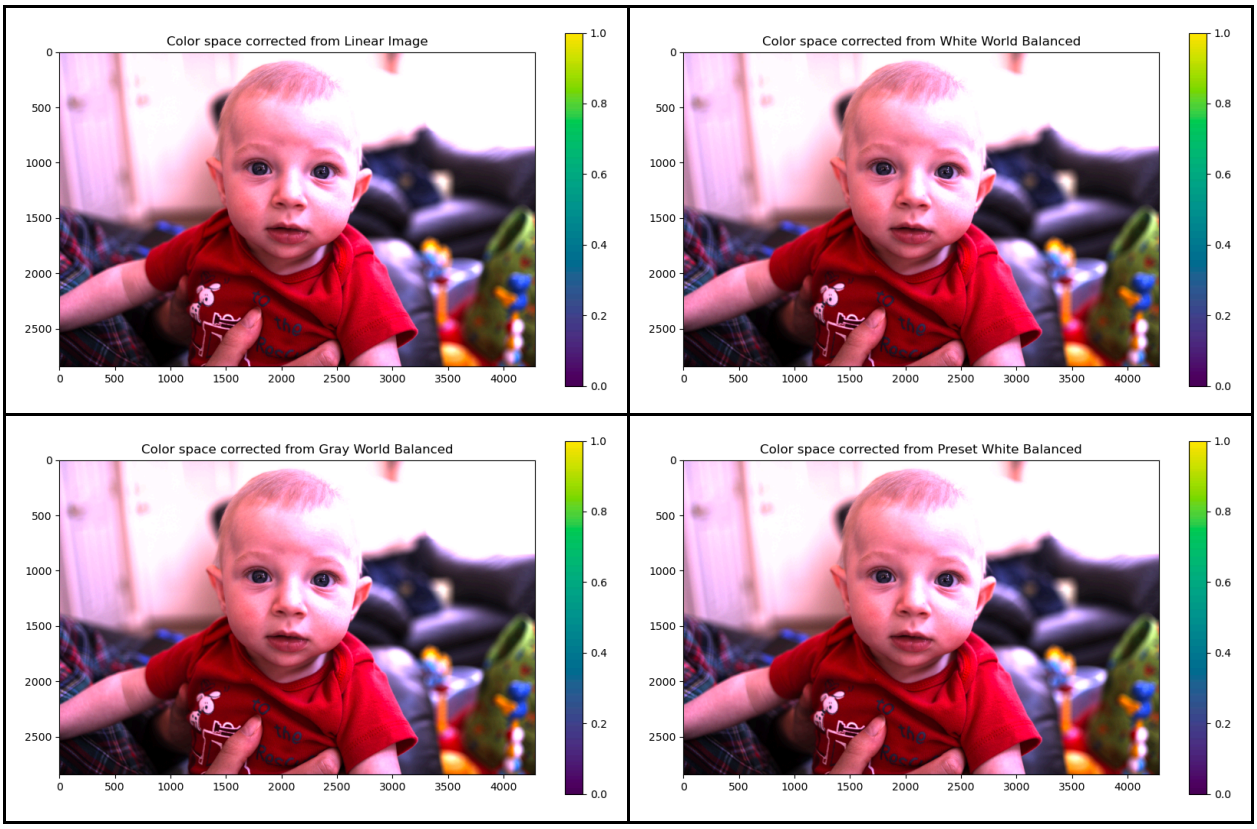
Demosaicing

Since interp2d is depreciated, I used another function called *demosaicing_CFA_Bayer_bilinear* from the library colour_demosaicing *demosaicing_CFA_Bayer_bilinear*

The resulting images right after demosaicing are shown in the following figure.

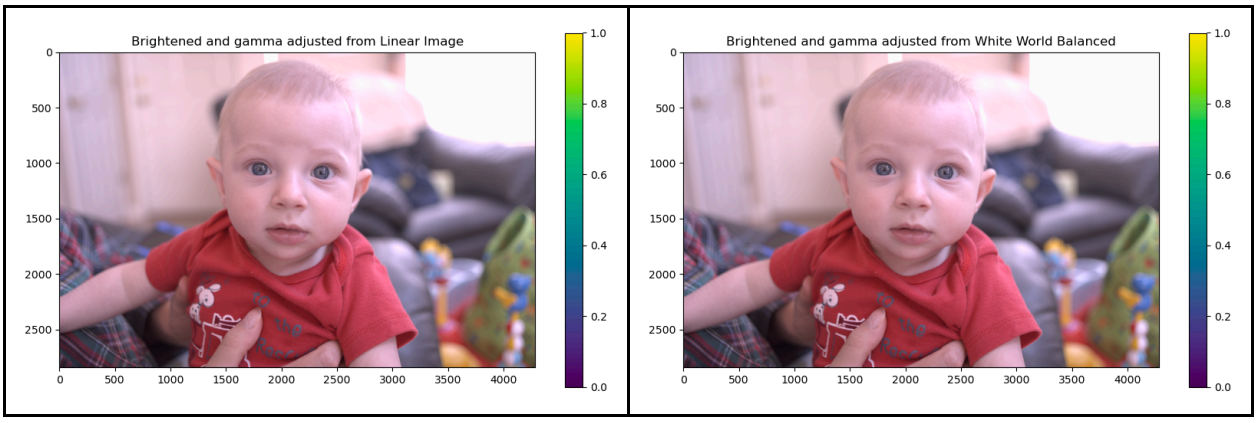


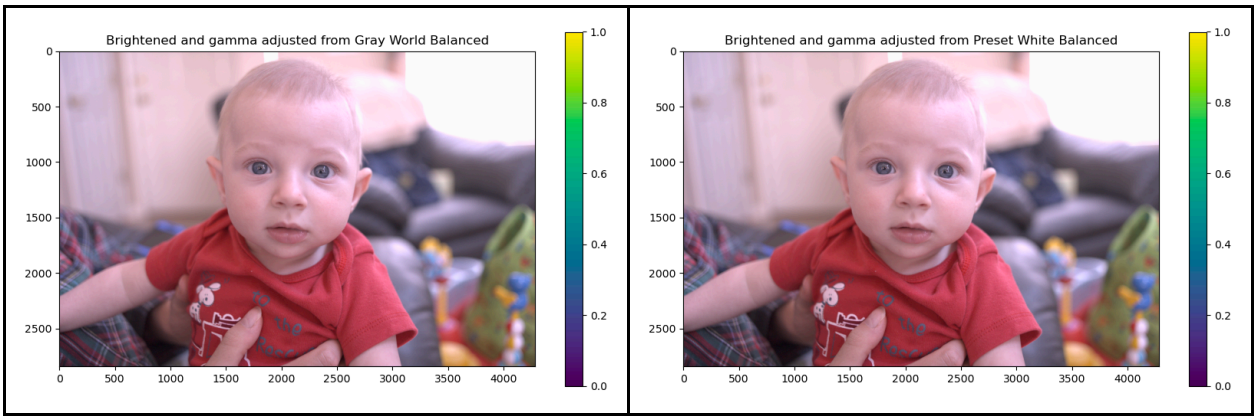
Color space correction



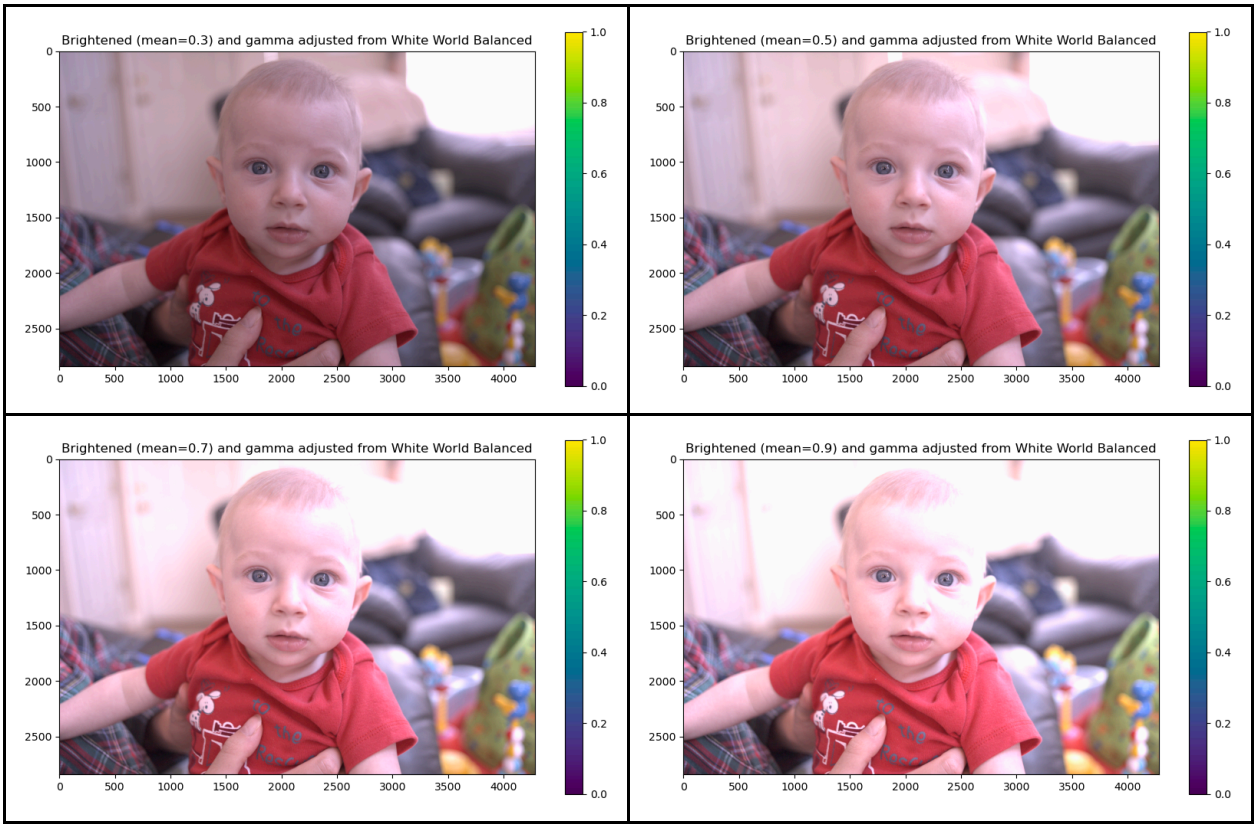
Brightness adjustment and gamma encoding

The resulting images with post brightening mean is set to 0.5.





Experiment with many percentages



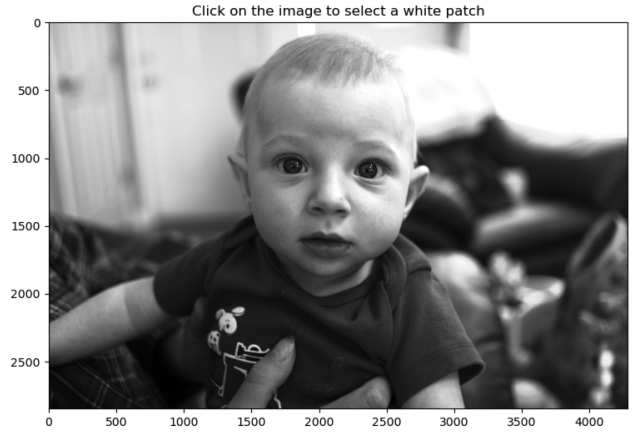
With post brightening mean 0.5 (top right) looks best to me. Not too dark, and no unnecessary glares.

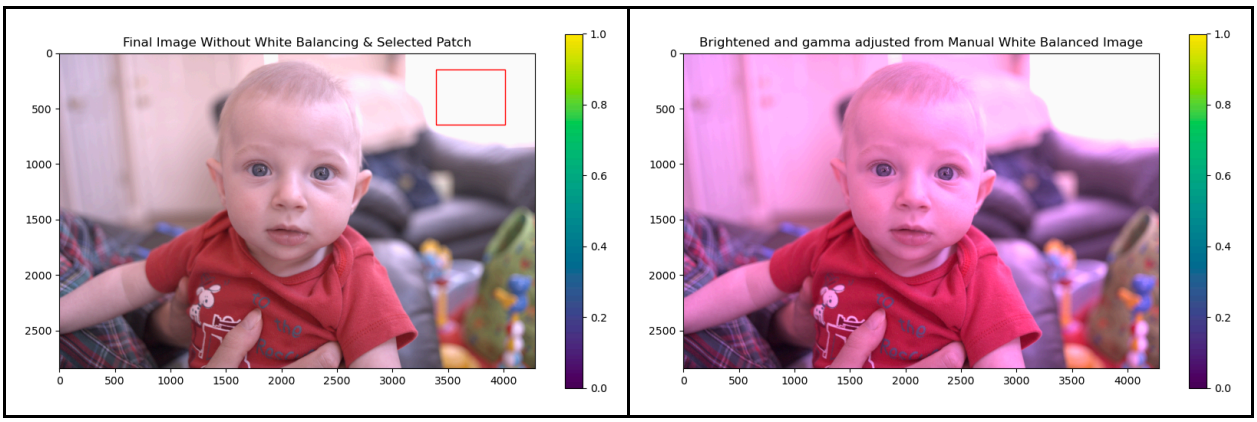
Compression

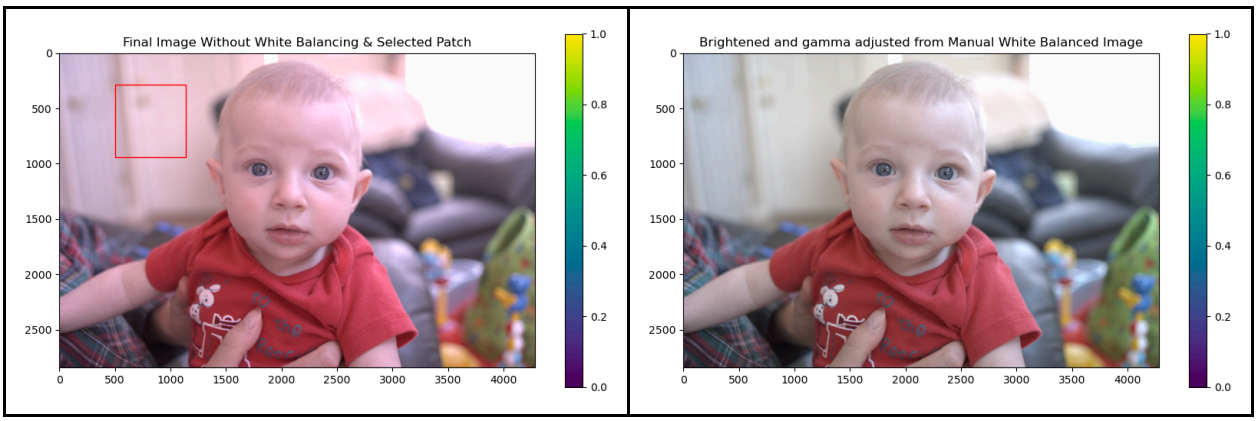
The difference between two images are not observable when quality is set to 90, however differences in images are observed when quality is set around 30-40.
Quality 50: images are not distinguishable.
Quality 30: wavy pattern in the background.
Quality 20: wavy pattern became obvious and flat areas due to sparse quantization.
Quality 10: Observable quality drop. Loss of color details and details of the object surfaces. For example, rectangular shadows on the door and speckles on the baby's face are invisible.
Quality 5: Cartoonish appearance, loss of high frequency details. Significant change in color tones.

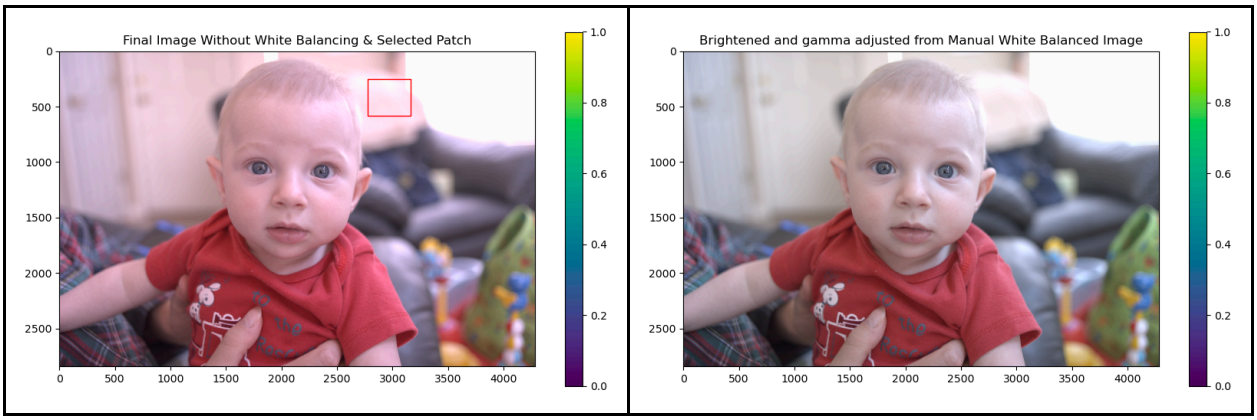
The following figure shows images with compression ratio 95, 30, 10 and 5, respectively.

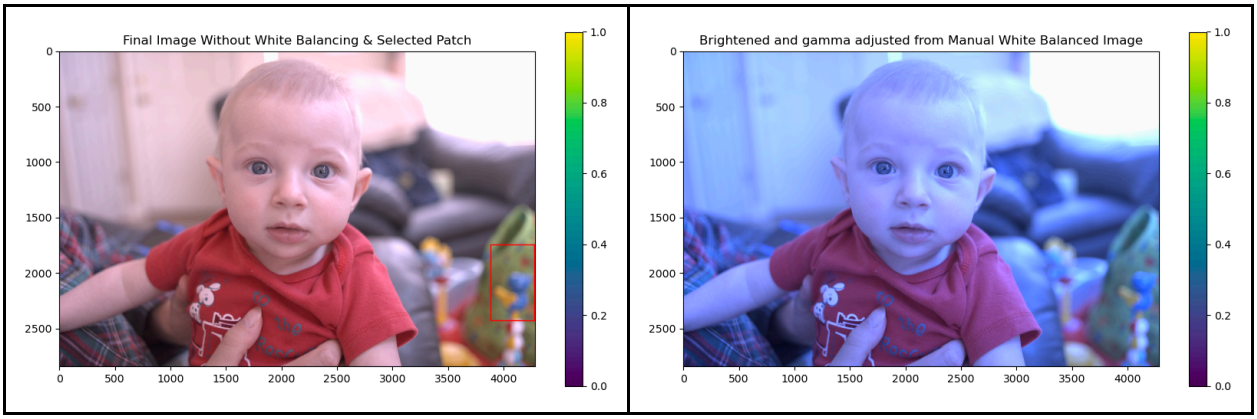
1.2 Perform manual white balancing











The images on the left are the final image without white balancing, and the images on the right are the final images with manual white balancing.

The greenish effect is reduced in the first two manually white balanced images. I used a patch containing the window I assume is the whitemost part in the image.

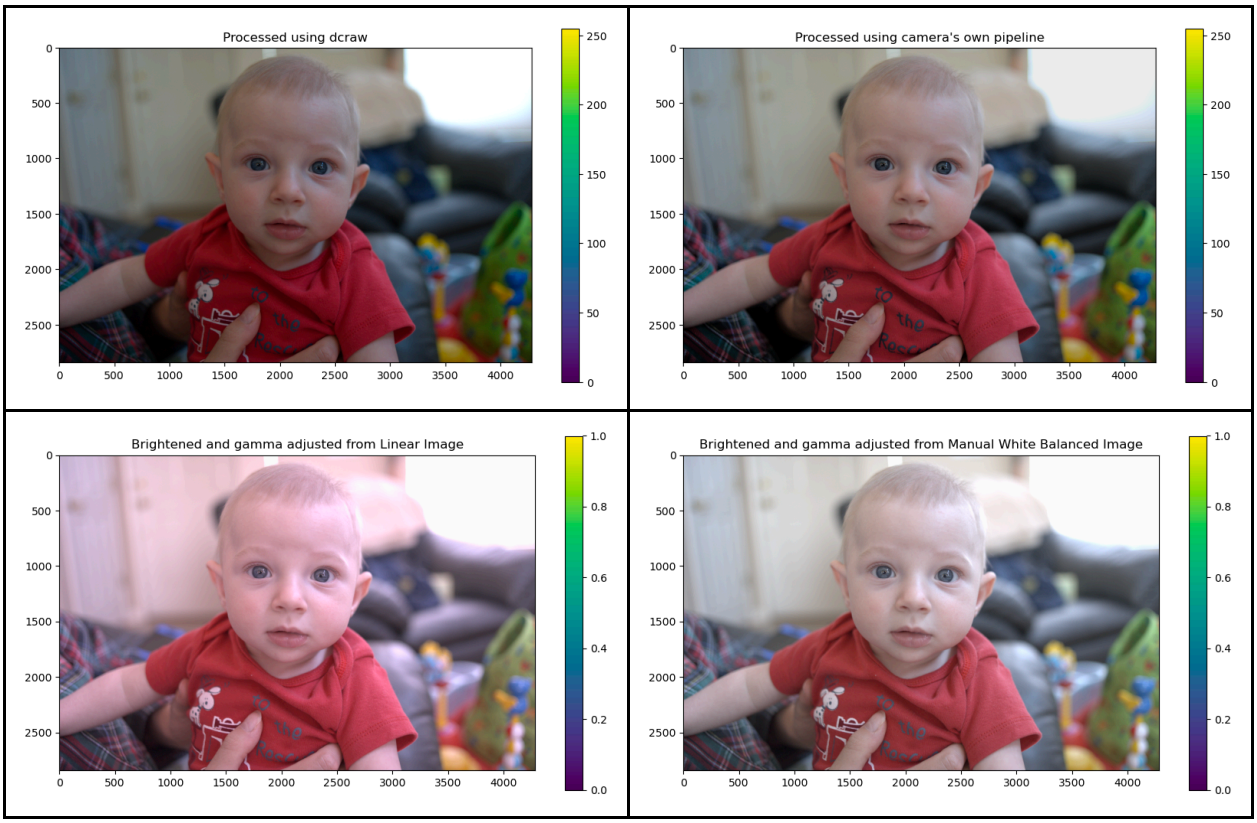
In the third image, I selected a patch from the wall, the resulting image became more red.

I think the first two patches work best, because the baby's face looks more natural, less flushed.

The last image is to show how the colors change in the image when a patch with a green object is used.

1.3 Learn to use dcraw

```
dcraw -v -w -q 3 -o 1 -b 1.2 -c ../data/baby.nef > ../data/dcraw_output_1.ppm
```



The colors in the dcraw output and the output of the camera's own pipeline looks very similar. The camera pipeline's output seems more natural and the brightness is slightly higher. The final image from the image processing pipeline I implemented looks best to me. Because the colors are more vivid, details are more apparent (I think due to the higher contrast). Also, the final image is less grayish and cloudy.

2 Camera Obscura

2.1 Build the pinhole camera

screen size: w:30 cm, h:21 cm

focal length: 28 cm

field of view: ~70 degrees.

2.2 Use your pinhole camera

Following are the captures with pinhole size 3, 4 and 5 cm.

With a 3 cm pinhole, the light exposure is not adequate to capture a high contrast image.

With a 4 cm pinhole, a good sharpness and color information is captured.

With a 5 cm pinhole, color information is good however, the image starts to be blurry.

Here are 2 more scenes with the same pinhole sizes:

2.3 Bonus: Camera obscura in your room

Following are the window setup, different pinholes, pinhole and corresponding captured images.