Processing Image Raw Data to Create an HDR Image

Students:

Pengcheng Ding (desmond)

Michael Lyons (mnlyons)

Sukaynah Almutawa (sukaynah)

Tami Gabriely (tami)

Problem Statement:

Using data from a single raw image, we shall be able to create an HDR image. An HDR image is usually created through the process of composing several images of the same object taken with different exposure levels to produce an image that has detail both in its brightest values as well as the shadows. Our purpose is to create the same effect, but only using one raw image. This can be especially useful for photographs of moving subjects or when the photographer is not able to hold the camera in the same position to take the three photographs (which is necessary for the usual approach for creating HDR images).

Definition of Success:

Success for us means to be able to output an image that can represent a high dynamic range of values from the environment. This means that the image will show detail in the brightest and darkest areas, without producing areas that are all black or all white, something that happens to low dynamic range images. Assuming that our assumptions (see below) are met, we will be able to output this kind of image using a single raw image as the input.

The Data:

The source of the data is Tami's Camera, a Canon Rebel T3i or a borrowed Canon 6D.

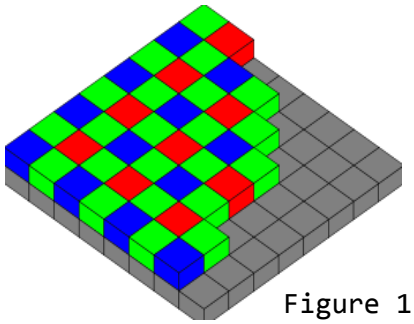


Figure 1

The data that is being used for the project consists of raw images taken with the Canon camera. Raw images contain direct raw data from the camera sensor. Although it can be slightly different or varied depending on the camera manufacturer, in general, direct data from the sensor makes use of the bayer pattern. This bayer pattern a color filter array that is 50% green, 25% blue, and 25% red (shown in figure 0). This preserves most of the information about the captured image, which includes information about the brightness and color at different pixels.

Different camera manufacturers make use of their own proprietary formats, which are different and not compatible with each other. This makes it difficult to create a program that can read all the different formats using the same code. Fortunately, Adobe wrote another raw image format called DNG (for Digital Negative). We can use the free and open Adobe DNG converter to convert any raw image to DNG, which is the format that we will be using to read in our images.

Prior Knowledge and Assumptions:

1. The inputs are images in a RAW format.
2. Our very first assumption was that we could actually find a way of reading a raw image into a workable environment. It is important that we can access the raw data of the image to be able to manipulate it fully.
3. An important assumption about our data is that the images were shot outside in the sunlight. This is an important assumption because of the nature of HDR. The reason photographers create HDR images is because they would like to be able to show details in the dark and bright areas of the image. These extreme values are generally created by harsh sunlight. When the sun is bright, it creates very bright areas from the light and very dark areas from the shadows. If this high range of values does not exist in the image, HDR is not necessary, and our program will not necessarily create pleasing images. Essentially, this assumption is saying that we only use images that "need" HDR, or images that otherwise would not show details in shadows and highlights.\
4. Another reason that we assume our images are shot outside in daylight is that the light is sufficient. Images shot indoors or outdoors in the dark tend to get noisy because of the limited light. Trying to brighten dark areas of images shot indoors results in a lot of grain in the shadows, which is something we want to avoid.
5. An additional assumption is that the image has a reasonable brightness range. In other words, we are not dealing with a photo that is totally over or under exposed throughout the whole image. The reason for that is that a photograph that is extremely over or under exposed lacks sensor information in the bright or dark values, respectively. Without sensor information in parts of the image, we would not be able to create an HDR image.

Sub Tasks and Approach (Techniques):

Read data from Raw image

1. Converting the raw images (in this case, the format is .CR2, which Canon uses) to .DNG.
    - To do this, we used the open source Adobe DNG converter. It was important to make sure that we change the preferences so that the output is uncompressed and not linear/demosaiced (the converter gives the option to make the output linear and depressed).
2. Linearizing the raw image

- Mapping the values of the raw array through a linearization table from DNG metadata to workable 10-14 bit values.
- Obtaining the black level value and saturation level value from the image meta info
- Linearizing and normalizing the data values to range [0, 1] by applying an affine transformation to image pixels.

3. (Optional) White balancing the image
    - Making three color masks that one of them set one channel's multiplier to 1 and the other two multipliers to what the camera recorded at the time of shooting
    - Multiplying every pixel in red-location of the image by the red multiplier and every pixel in blue-location of the image by the blue multiplier. To be able to do this, we applied a dot-multiplication with a mask of red scale values and blue scale values.
    - Inverting the multiplier values and scaling them again to make green multiplier equal to 1.
4. Demosaicing the image to generate a viewable RGB image
    - Scaling the entire image so that the max value of the image is 2^16 in order to obtain an unit-16 input
    - Applying a demosaicing function that's built in MATLAB to generate a RGB image variable
    - Scaling the image values back to range [0, 1] to get a viewable RGB image
5. Converting the RGB image to the correct color space
    - Applying a 3x3 matrix transformation to each of the pixels so that the color basis is recognizable by the monitor. We used a sRGB to XYZ colorspace transformation matrix from Bruce Lindbloom's website[2].

Create an HDR image

6. Creating multiple images with different brightness levels
    - We did this because we could pretend we have some images with different exposure values so that we could combine them into a HDR image
    - Converting the RGB image to a grayscale image
    - Scaling the mean luminance of the gray image so that we end up with six images with brightness levels from low to high (we chose six because we've tried working with more images but six images were the best in terms of running-time and final results)

7. Building a user-friendly interface which allows users to select image that they want to work with
8. Making a HDR image
    - Using MATLAB built-in function to create a HDR image from the six images with different brightness values
    - Experimenting with different sets of relative exposure values as input for the function we used to make HDR image
    - Adjusting the saturation and brightness of the image while we making HDR, which was also implemented as track bars in the user interface
9. Exporting the image to a viewable format (we used PNG)

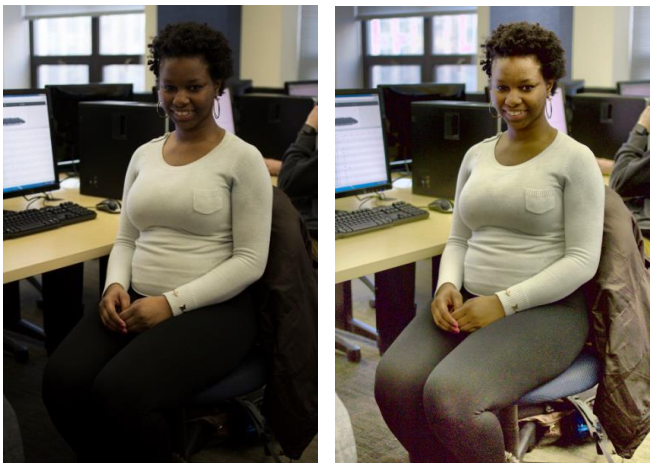


Figure 2. Original on left, our output on right.

Figure 3. Original on top, our output on bottom.

Discussion of Final Results:

At the end of our project, our code works about as well as hoped. We were able to successfully read the raw image files from our camera database and convert them to files that could be more easily manipulated. We were also able to successfully take these converted image files and give them high dynamic range, in essence taking the brightest and darkest areas of the image and giving them an aesthetically pleasing dynamic range.

As expected, images in an indoor environment did not respond as well to the program, leaving noise in the darker areas. As seen in the images in figures 2 and 3, that were shot indoors with poor lighting, the dark areas of the images are filled with noise. Although the output does have a high dynamic range, the images are too grainy. Also, images that were initially too dark will return an obscenely noisy image. As seen in figures 4 and 5, the output image is very noisy and has a lot of color

Figure 4. Original on top, our output on bottom.

noise as well. Figure 5 additionally has a pink artifact that we observed in images that are too bright. Images that are not in need of the HDR process result in a similar image with additional noise and a slight and arbitrary exposure difference, as seen in figure 6. This image is not terrible, but also isn't very necessary, since the original already shows a fairly high dynamic range. Images that were initially too bright also did not turn out very good, shown in figures 7 and 8. The areas of the bright images that had absolute white values turned pink when manipulated in MATLAB which resulted in an image that was very different from the original. All of these shortcomings were expected based on our assumptions.

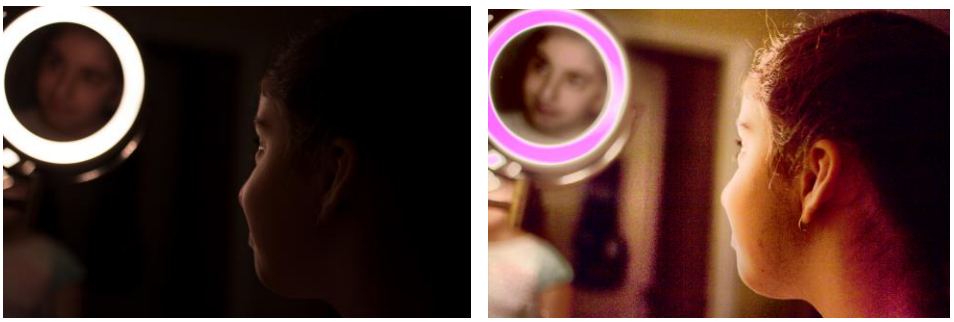


Figure 5. Original on left, our output on right.

Figure 6. Original on left, our output on right.

Figure 7. Original on left, our output on right.

Figure 8. Original on left, our output on right.

One imperfection that we didn't account for was acceptably bright images with small regions that were too bright and had no range. Essentially, bright white space in an image does not respond well to the function, and returns in the final image with a pink tint instead of plain white. This can be seen in some of the images of the runners at the end of the report. The

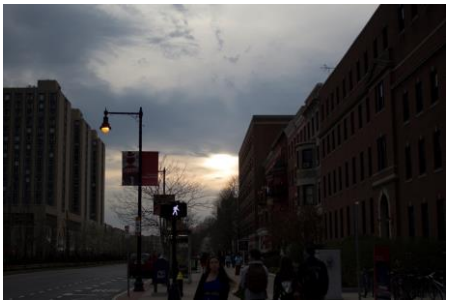
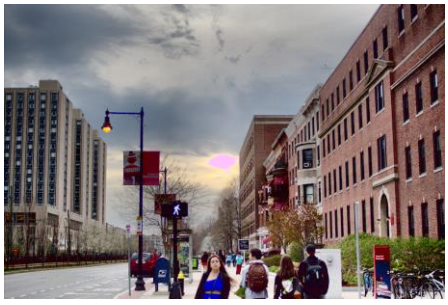
source of this bug is still undiscovered. We don't know exactly when in the process this happens, but it is visible in the six separate images even before the process of making the HDR file. This was still visible even when we tried changing the brightness of the images using different means (such as the imAdjust() function).

We feel this was a successful approach because we made good use of built-in functions (such as makehdr and demosaic) that were already available to us instead of trying to build unnecessary functions from the ground up. The use of these functions made the coding of our project less buggy, but slightly more restricted. However, the functions we used were perfectly suited to what we intended to use them for, giving us a successful final result.

Below are a few examples of successful outputs (input on left, output on right). These are successful because they show a range of values in the dark and the bright areas of the image.








Possible Future Work:

Few ideas for taking our program forward:

- Introducing an interface with more options for the user to interact, such as determining the specific exposure value for each subimage to pass to the `makehdr()` function in MATLAB.
- Decreasing noise levels in images affected by an increase in unnecessary noise, such as indoor images and overly bright/dark images, by using a mean image filter.
- Finding the source of the pink tint bug in bright white space and correcting it, or correcting post-HDR by comparing pink areas in the final image to the original and reverting them back to white if they don't fall within a certain threshold.

Sources

1. http://users.soe.ucsc.edu/~rcsumner/rawguide/RAWguide.pdf
2. http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html