6.815/6.865 Digital & Computational Photography

Problem Set 1: Demosaicing and White Balance

Due Tuesday, February 26 at 7:00pm

Introduction

Digital cameras don't see the world as we do. To get from the raw sensor readings to the final image that we end up viewing, a number of steps must be performed. Two of the most important ones are *demosaicing* and *white balance*. This assignment explores techniques for these operations. Upon completion of this assignment, you'll have a basic knowledge of how RAW conversion works. But before we get into the details, let's go over some logistics.

In each problem set, there will be parts that are labeled as **6.865**. If you're registered for 6.815, you can skip them. Of course, you may submit solutions if you want, and we'll give you feedback, but they won't count towards your grade.

Most problem set turnins will include both programming and written components. You will generally be asked to submit a set of MATLAB files and a PDF file containing answers and supporting images for each written question. We recommend LATEXfor typesetting.

You are encouraged to discuss the assignments with others in the class, whether it's in person or on `6.815-discuss`. However, you should do coding and writing individually.

Demosaicing

In the accompanying files for this assignment, we have provided several Bayer pattern images from a Nikon D70s digital camera in PGM format (16-bit linear gamma). These were converted from the original NEF files using Dave Coffin's **dcraw** program[1]. If you'd like to play with your own digital camera files, try it out. In fact, we encourage you to use your own photographs if possible (you will need a camera with RAW output capabilities if you want to do your own demosaicing). We used `./dcraw -4 -d -r 2 1 2 1` on our NEF files to generate the provided PGM files.

Let's start by loading up one of these files into MATLAB and looking at the layout:

```
R = imread('signs.pgm'); imtool(imadjust(R, [], [], 1/2.2));
```

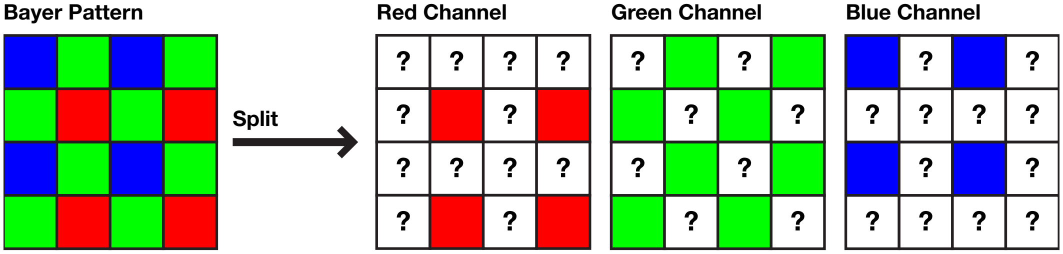
The call to `imadjust` compensates for the linear gamma of the PGM file. If you zoom in on the image closely enough to see the individual pixels, you'll see the Bayer pattern.

---

[1] Freely distributed at `http://cybercom.net/~dcoffin/dcraw/`.

Problem 1 (6.815/6.865)

Implement a MATLAB function `demosaic_linear(R)` that produces a full-color RGB image from a Bayer pattern image using linear interpolation.



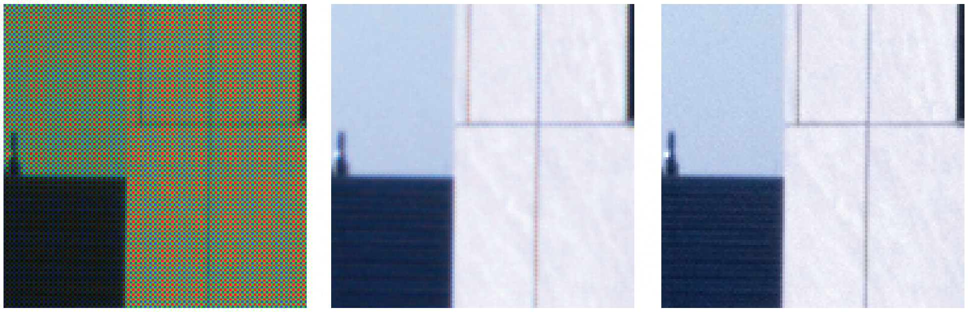
The above image shows the Bayer pattern for the provided PGM files (note that the top-left pixel is blue). Basically, your function should split the given image into three channels and fill in all the question marks in each channel. You may ignore the one-pixel border around the image. The PGM files are large, so you may wish to crop the images for testing.

This is a simple method and is not without flaws. Experiment with the various images and see where the technique fails. Specifically, look at edges and thin lines.

In your writeup: copy and paste several examples (at least three) of these bad artifacts. Don't paste entire images; crop out portions that highlight the issues.

Problem 2 (6.815/6.865)

Implement a MATLAB function `demosaic_median(R)` that produces a full-color RGB image from a Bayer pattern image using median interpolation. Try using a 5×5 window.



This shows the original Bayer pattern image, the result of `demosaic_linear(R)`, and the result of `demosaic_median(R)`. Median interpolation was performed by converting the result of `demosaic_linear(R)` to YCbCr, filtering the chroma channels (Cb and Cr) with `medfilt2`, and reconstructing the luma channel (Y) using the original Bayer pattern image. The details of this technique will be covered in lecture.

In your writeup: show the result of median interpolation on the examples that you chose for Problem 1. In addition, explain a situation (and provide an image) in which median filtering *introduces* undesirable artifacts.

White Balance

The white balance operation corrects for color casts induced by different illuminants. Many digital cameras offer an automatic white balance operation (and they often don't work very well). You'll be experimenting with two simple techniques that assume that white balance can be achieved by independent transformation of each color channel (this is known as von Kries adaptation).

Let's start by taking a look at some of the TIF files. These are 16-bit linear demosaiced versions of the PGM files (done with dcraw, which uses AHD demosaicing[2]). As before, you can view the images by applying gamma correction:

```
I = imread('hall.ppm'); imshow(imadjust(I, [], [], 1/2.2));
```

Usually, color correction is done before gamma correction, so for these problems, you should defer the `imadjust` operation until it is time to view the image.

We encourage you to try using some of your own photos (but it's okay to use the provided ones). For best results, you should shoot in RAW with your camera locked on daylight white balance and convert to 16-bit linear using dcraw. Ask us if you need help with this.

Problem 3 (6.815/6.865)

Implement the following white balancing techniques in MATLAB:

`whitebal_max(I)`: Assumes that the brightest pixel in the image should be white. For each color channel, find the pixel p with the maximum intensity, and then scale the entire color channel so that p is maxed out (with 16-bit linear, 2^{16}). This approach can be unreliable if pixels are clipped, so you should avoid choosing any p that is already maxed out in *any* channel.

`whitebal_gray(I)`: Assumes that the average color in the image is a shade of gray (this is generally known as the gray world assumption). What shade of gray should you pick? Well, this is flexible, but in general, you should try to maintain the brightness of the original image. For instance, if the original image is very dark, you want the result to average out to a dark gray. Try out different strategies for this and choose something that seems to produce good results.

In your writeup: Compare and contrast the two techniques that you implemented. Try to answer the following questions:

- In what sorts of photos would one strategy be preferred over the other?
- How sensitive are they to slight changes in the scene (for instance, if someone with a pink jacket enters the corner of frame)?
- What sorts of scenes would cause the strategies to fail, and how badly?

We're not expecting a paper here, but you should write about two or three paragraphs and support your discussion with images generated by your code.

---

[2] Paper can be found here: `http://citeseer.ist.psu.edu/hirakawa05adaptive.html`.

Problem 4 (6.865)

This problem is for 6.865 students only.

Most white balance techniques make the assumption that there is only a single type of illuminant in a scene. This is often not the case. For instance, our classroom has fluorescent lamps and windows that let in natural sunlight. This often creates difficult situations that photographers must fix using color gels or supplemental lighting.

For this problem, you will implement a local white balancing technique from ECCV 2004 that tries to compensate for differing illumination: "Color Constancy Using Local Color Shifts" by Marc Ebner. Fetch the paper here:

`http://www.springerlink.com.libproxy.mit.edu/content/lu5ydgvb0hlrcwdc/`

You should be able to access this link with MIT certificates.

Write a function `whitebal_local(I)` that implements this technique. Don't worry about Section 3 of the paper; it just describes a potential hardware implementation. The "local space average color" images are simply blurred versions of the original images, which you can compute using `imfilter` and `fspecial` (read the MATLAB help files on these functions). You just have to look at Section 4 (and really, just the end of the section).

In your writeup: Show a few images generated by your implementation. Try out different types of lighting. What sorts of scenes does this technique work well on? When does it fail?

Submission

Like the previous assignment, you should assemble a ZIP file that is named after your Athena login. Make sure this file contains:

- A PDF file with answers to your written questions and your results. *In general, you should try to make this file as self-sufficient as possible.* In other words, we shouldn't have to look at your code to evaluate your results. Please don't tell us to run something unless it's absolutely necessary. We will look at your code to make sure you did the work and assign partial credit if you did something wrong.
- Your MATLAB code:
  - `demosaic_linear.m`
  - `demosaic_median.m`
  - `whitebal_max.m`
  - `whitebal_gray.m`
  - `whitebal_local.m` (6.865 only)
- Any images (other than the provided ones) that might be necessary to run your code.

All submissions are due on the Stellar website by Tuesday February 26 at 7pm.