Florian Kordon, Thomas Köhler, Vincent Christlein, Mathias Seuret, Mathias Zinnen

Demosaicing & High Dynamic Range

Deadline: 2023.11.24

Download the Data

Download the data. Note that in a few days, some extra photos to play with will be uploaded as well.

1 Investigate Bayer Patterns

In order to get familiar with the Bayer patterns, have a look at Image IMG_9939.JPG. You can see red, green and blue pens, on a white background. The file IMG_9939.npy contains a numpy array storing the raw sensor data – load it and try to find out how the Bayer filter is. It is *not* like the one shown on the slides.
Hint: keep in mind that the green channel typically has higher values, even on non-green pixels.

2 Prove that Sensor Data is Linear

Images IMG_3044 to IMG_3049 contain the same scene captured with different exposure times. As a reminder, the amount of collected light is proportional to the exposure time.

These images had exposure times of respectively 1/10, 1/20, 1/40, 1/80, 1/160 and 1/320 seconds. Thus, each gets half the amount light as the previous one. Plot the average value of red, green and blue pixels in function of the exposure time (three curves!), and verify that you obtain straight lines.
Hint: You can load the raw sensor data with the following code:

```
import numpy as np
import rawpy
raw = rawpy.imread('IMG_3044.CR3')
array = np.array(raw.raw_image_visible)
```

3 Implement a Demosaicing Algorithm

Implement the simple demosaicing algorithm seen during the lecture, or a more advanced one. Apply it to IMG_4782.

The sensor data is encoded on 14 bits, so divide the values by the right constant to be in the usual range [0, 255].
Hint: no need to apply a median filter on U and V channels; it is slow and won't improve much the image. **Hint:** if it looks grayish, it's normal: you haven't applied some color balance yet.

4 Improve the Luminosity

As you probably noticed, the image you obtained after demosaicing is quite dark. Apply a gamma correction – I suggest $\gamma = 0.3$, and not to normalize using `np.min` and `np.max` but `np.percentile` with values of 0.01 and 99.99.

Evaluate at least one other curve than $y = x^\gamma$.
Hint: use the percentiles to normalize data in [0,1] as you would normally do with min and max (set negative values to 0, and values greater than 1 to 1), apply the gamma correction, and invert the transform to get back the the previous range of values.
Hint: here is how to use the percentile approach:

```
a = np.percentile(data, 0.01)
b = np.percentile(data, 99.99)
data = (data - a) / (b-a)
data[data<0] = 0
data[data>1] = 1
```

5 White Balance

The last step for processing raw sensor data is to apply white balance. Implement the gray world in your algorithm.
Hint: because of the multiplications, some values might become greater than 255; clip them to this maximum.

6 Initial HDR Implementation

This exercise is the initial implementation of HDR images, as seen during the lecture. You will find on StudOn a set of photos (from 00.CR3 to 10.CR3). For each of them, the exposure time was half the one of the previous photo.

Combine them to produce HDR raw data. Then, apply the demosaicing algorithm and the white balance. Decrease the dynamic range by computing the logarithm of this data, and downscale it to the [0, 255] interval. Then save the resulting image.

You should obtain the same result as what is shown in the slides.

7 iCAM06

Implement the iCAM06 HDR method, as explained on the slides. Try different settings to find the ones you like the best.
Hint: bilateral filters are slow, so while developing your solution, use at first very small kernels to minimize the run time.

8 Win

.. or at least try to!

Implement a demosaicing function named `process_raw` which takes two parameters:

1. The path to a raw file (CR3 format)
2. The path to wich the resulting JPG image has to be saved

Hint: you will be judged based on the quality of your image, so when saving it to the JPG format, set a high quality setting, such as 98 or 99, not 75!

9 Additional Exercise

This exercise has to be done only if you go for the option 2 described on Slide 4 of lecture01.pdf.

Using raw data for producing HDR images offers the advantage of having a linear relationship between luminosity and pixel value. This is not true anymore when working on JPG images, as the camera introduces non-linearities.

The goal of this exercise is to produce HDR images from JPG images. For this, you have to estimate the camera's luminosity transformation curve $y = f(x)$, where y is the pixel value and x is the luminosity (as in slide 44 of the lecture). Inverting this function allows you to have the data in a linear regime again.

If you want to do this exercise, you will need additional data which is not on StudOn. Contact me to obtain some.