CMPE 264 – Project Assignment 1

You are to create a High Dynamic Range (HDR) system. The project is articulated in four parts. You will need to write your own code in OpenCV (recommended) or Matlab for Parts 1 to 3; you will use pre-packaged functions in Part 4.

Part 1: Camera radiometric calibration

You will need a camera to take pictures. **You must take the pictures – not download them from the web.** Your camera must allow you to control the exposure time T and gain G independently. For example, you could use your smartphone's camera, but you will need to install an app that gives you the ability to control exposure parameters. If you are using an iPhone, you could use Camera+; for Android phones, you can try the Camera FV-5 app.

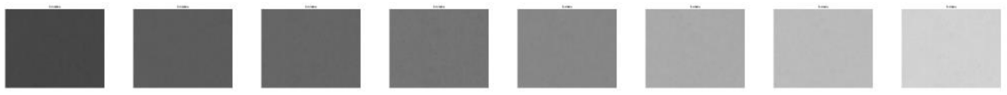
It is very likely that your camera will not produce a linear response (unless you are able to obtain images in raw format; this is usually not possible with smartphones). This means that there will be a non-linear mapping between the recorded data $E\cdot T\cdot G$ and the output brightness (the pixel values you work with). More specifically: if the camera were linear, you would measure $B=c\cdot E\cdot T\cdot G$, where c is an (unknown) proportionality factor (following the derivation in the slides, $c=2^N/Q_{sat}$, so that the data in the top quantization bin is mapped to the largest output value 2^N.) However, you cannot measure B directly; instead, the camera gives you a value $B'=f(B)$, where $f(\cdot)$ is an unknown non-linear function, which changes from camera to camera.

For HDR to work well, you need to reverse-engineer the function $f(\cdot)$ and invert it, obtaining an approximation of the true (linear) brightness $B=f^{-1}(B')$. This is called *radiometric calibration*. There are many ways to estimate $f(\cdot)$. In the following I describe a simple method that gives reasonable results. You can use this method or any other method, but you will need to explain it carefully in your report. Note that you will need to radiometrically calibrate each color channel of your camera (i.e. find $f_R(\cdot),f_G(\cdot),f_B(\cdot)$).

You will need to take multiple pictures of a flat surface with uniform radiance – say a flat sheet of white paper – where each picture is taken at different exposure time T (leave G fixed – e.g. set G=100 ISO). Make sure that the illumination on the paper doesn't change while you are taking the pictures. Then crop out a central portion of each image and (for each color channel) compute the average value in this central area. For each color channel, you will obtain a set of values $\{B'(T_1), B'(T_2),\ldots, B'(T_n)\}$ where $\{T_1,T_2,\ldots,T_n\}$ are the exposure times you used.

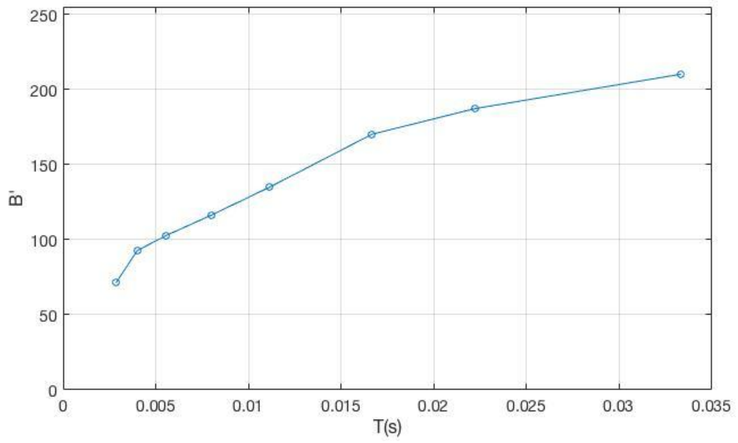
Make sure that pixels are not saturated (any pixel that takes a value of 255 can be considered to be saturated) – if one image has saturated pixel, that T_i is too large and you shouldn't use it. Also, make sure to choose a set of exposure times and a gain G such that you span a as large as possible set of (non-saturated) pixel values $B'(T)$. Ideally, at the shortest exposure time T_1, $B'(T_1) < 40$, while at T_n, $230< B'(T_1) < 255$.

This are the green channel values for the cropped images I took (not ideal: I should have used more exposure times to span a larger set of B' values):

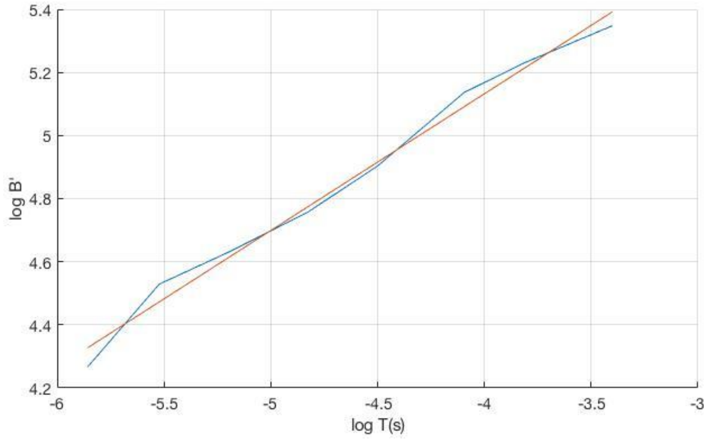


<ins>We will make the simplifying assumption that $B'=f(B)=B^{1/g}$ for some unknown value g</ins>. This is usually a decent first-order approximation.

Our task is to find the parameter g, so that we can retrieve the linear brightness B as $B=(B')^g$. Note that what you are measuring is $B'(T)= f(B(T))= f(c\cdot E\cdot T\cdot G)=K\cdot T^{1/g}$ where $K=(c\cdot E\cdot G)^{1/g}$ is a number that is independent of T. Here is a plot of the measured B' as a function of T with the images I took:

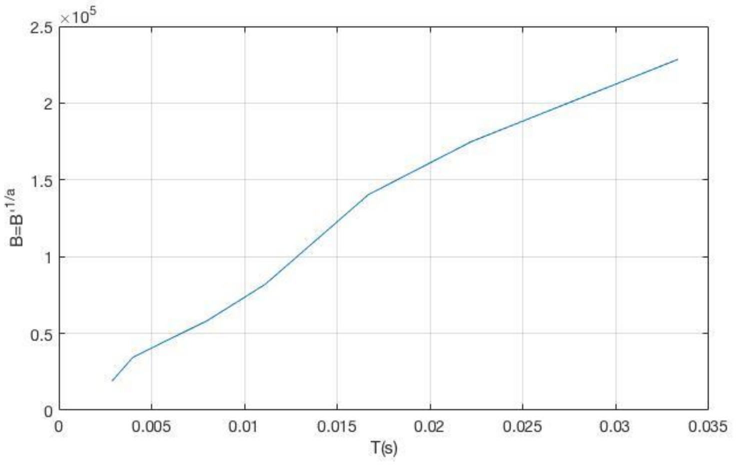


You need to estimate (regress) the parameter g from $\{B'(T_i)\}$. Here is a possible way to do it; feel free to use any other non-linear regression technique (but document it in your report). If I take the logarithm of B', I obtain:
$\log(B'(T)) = \log(K\cdot T^{1/g}) = \log(K) + (1/g)\cdot\log(T)$. Hence, $\log(B'(T))$ is an affine function $(b+a\cdot\log(T))$ of $\log(T)$, where $b= \log(K)$ and $a=1/g$ are constants. I plotted $\log(B'(T))$ against $\log(T)$ below (blue line) below.



Simple linear regression allows you to find the parameters b and a=1/g and thus to create a linearized version of your data (that is, B'^{g}). The orange line above shows the regressed line b+a·log(T), with a=1/g=0.97 and b=6.866 (which I found by linear regression).

Plotting B'^{g} against T should result in a linear function:



Not perfectly linear, but a decent approximation. Remember to do this for each color channel.

Deliverables:

1. The pictures you took, labeled with their exposure time and gain.
2. The plots B'(T) as a function of the T you used for all three channels
3. The values of g for each color channel (they will be similar but typically slightly different)
4. The plots of B'^{g} (T) as a function of the T you used for all three channels

Part 2: Acquire a picture stack

You need to take a stack of 3 images of a high-dynamic (high contrast) scene. I suggest looking for an outdoor scene with sunlight and shadows – it may be difficult to get a high dynamic scene indoors. **It is critical that the pictures be spatially registered with each other!** You will need to stabilize your camera somehow – e.g. with a tripod (you can buy tripods for smartphones for $10 or less). I suggest taking a stack with multiple pictures at different T with fixed G. You can then decide which 3 pictures to extract from this stack.

To choose which images to select, you need to look at the image values (I suggest looking at their histograms). The first image should be at "optimal exposure" (exposed to the right) – i.e., taken at the largest T that gives no saturated pixels. Then, choose two more images at larger exposure times a_1·T and a_2·T with a_1, $a_2 > 1$. You should select the values a_1 and a_2 based on the image contrast – e.g. a_1=5 and a_2=10.

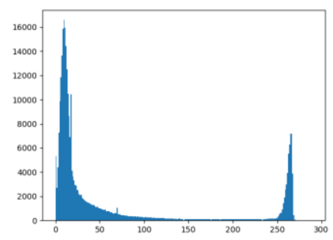
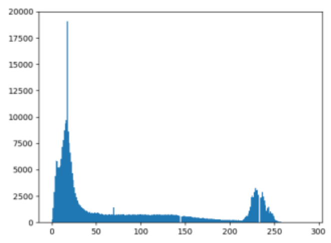
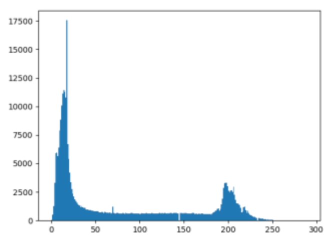
At this point you have a stack of three images at exposure times T, a_1·T and a_2·T. Make sure to linearize their color values as discussed in Part 1. [Note: the original images are probably represented in memory as uint8; after linearization (Part 1), you will need to convert them to float.]

Deliverables:

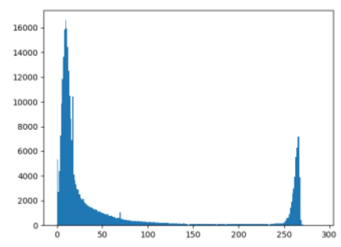
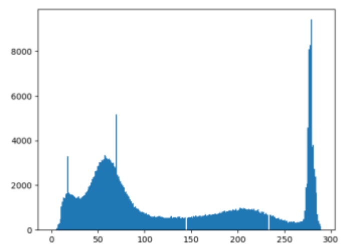
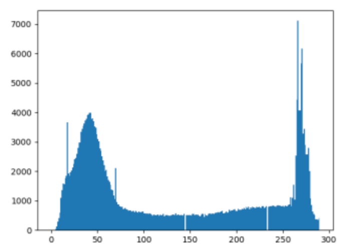
- The full stack of pictures you took with their exposure times and gain. Make sure to indicate which are the 3 pictures you selected
- The histogram of the values $B'^g(a_i \cdot T)$ for each color channel for each of the 3 pictured in the stack. (In this notation, i=0,1,2, and $a_0=1$). When you compute the histograms, use a set of 25 uniform bins between 0 and 255^g (which is the largest possible value of B'^g)
- For the second and the third image in the stack, also plot, for each color channel, the histogram of the values $B'^g(a_1 \cdot T) / a_1$ (for the second image) and $B'^g(a_2 \cdot T) / a_2$ (for the third image). Use the same bins as for the previous histograms.

Pictures

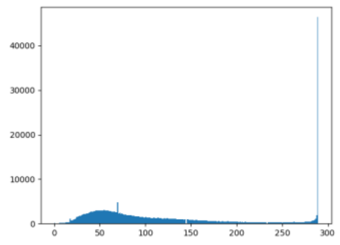
Histogram $B'^g(a_0 \cdot T)$

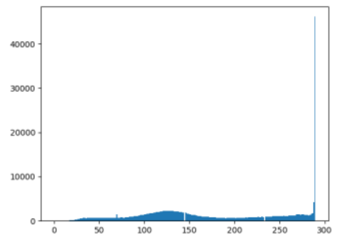




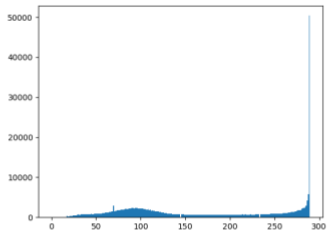
Histogram $B'^g(a_1 \cdot T)$

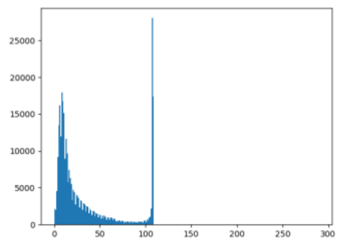
Histogram $B'^g(a_2 \cdot T)$

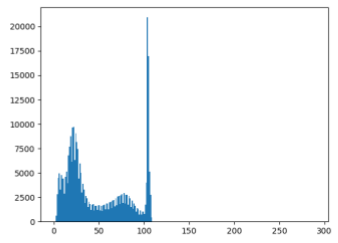


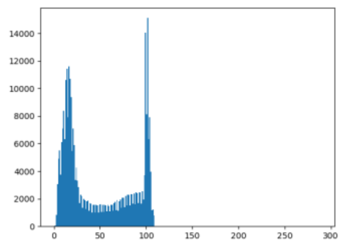




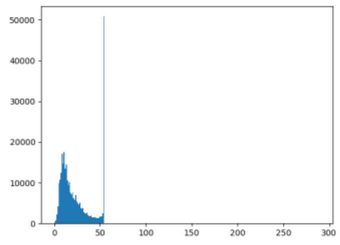
Histogram $B'^g(a_1 \cdot T)/ a_1$

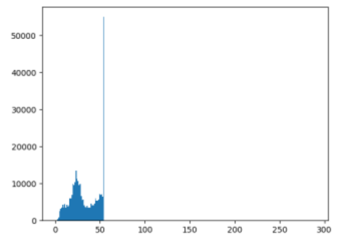


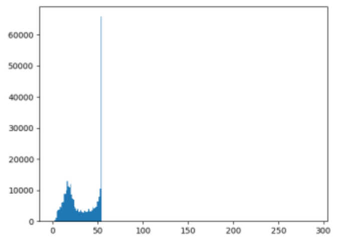




Histogram $B'^g(a_2 \cdot T)/ a_2$







Part 3: Create a composite image

You need to create a composite HDR image from your stack of three linearized pictures. You will compose the images using two different algorithms:

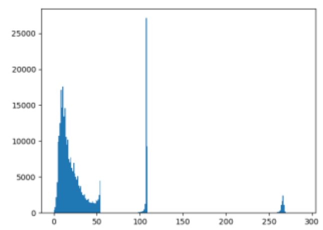
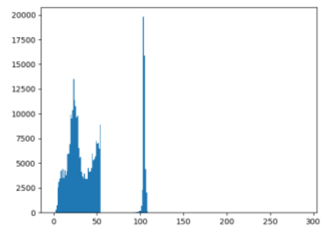
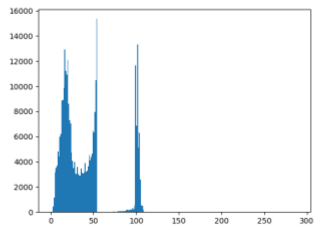
Composition Algorithm 1: see slide 21. For each pixel, use the pixel value of the image with largest exposure time such that the pixel is not saturated (i.e., $B'<255$ for that image). If you did things correctly, no pixel is saturated in the first image (exposure T); a set S_1 pixels are saturated ($B'=255$) in the second image (exposure $a_1 \cdot T$); and a set $S_2 \supset S_1$ are saturated in the third image (exposure $a_2 \cdot T$). Hence, if, for example, your pixel belongs to S_1, you would use the value $B'^g(T)$ from the first image. If it belongs to S_2, but not to S_1, you would use the value $B'^g(a_1 \cdot T) / a_1$ from the second image. And if it does not belong to S_2, you would use the value $B'^g(a_2 \cdot T) / a_2$ from the third image.

Composition Algorithm 2: For each pixel, use the average of the values in the images for which this pixel is not saturated. E.g., if the pixel belongs to $S_2 \setminus S_1$, you would compute the average of the value $B'^{g}(T)$ from the first image and of the value $B'^{g}(a_1 \cdot T) / a_1$ from the second image.

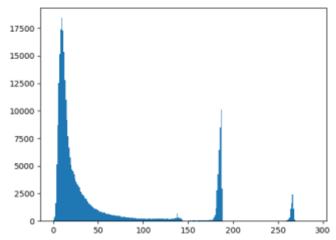
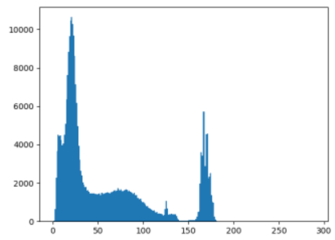
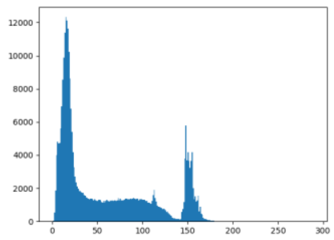
Deliverables:

- The histograms (for each color channel) of the composite image obtained using Algorithm 1 and of the composite image obtained using Algorithm 2.

HDR1 Histogram

HDR2 Histogram

Part 4: Reproduce composite image

You now have a composite HDR image, but if you again convert it to 8 bits, you are back to step 1. Instead, you should apply a non-linear function that basically amplifies values that are small and reduces values that are large. This can be a per-pixel function (e.g. logarithmic processing or histogram adjustment) or more complex, adaptive functions. For this part, you can use a tone mapping function from OpenCV (Tonemap class) or Matlab (tonemap function in the Image Processing Toolbox).

Deliverables:

- The tone-mapped composite images from Algorithm 1 and 2

HDR1

HDR2

What I am expecting from you

You will need to submit:

- A report that explains carefully all that you did and that contains all deliverables
- Your code – all project zipped. Harsimran needs to be able to compile and run the code. Include:
    - A README file describing:
        - OS, language, libraries used
        - Detailed instructions on how to run your code
    - Scripts that generate all of the graphs and images you are showing in your report.
    - If you think it will help us better understand your code or how to run it, you could also include relevant screenshots

You will be evaluated on:

- Correctness of your implementation of the project and results
- Quality of your report (including quality of your writing, images/plots included, and clarity)
- If your code does not compile/run as expected, you will be assessed a penalty