Uniwersytet Wrocławski | Instytut Informatyki

Computational Photography
Winter 2016-17

LAB 2: DUE 27 OCTOBER 2016

Task 1: Focal Stack (20 pts)

Compute an all-in-focus image from a focal stack. Three images of a focal stack are provided. Compute the magnitudes of their gradients for each pixel in each image. Fuse all three images by picking the pixel value with the highest gradient magnitude among the three. Plot an false color image with the image contributions.

Task 2: HDR (40 pts)

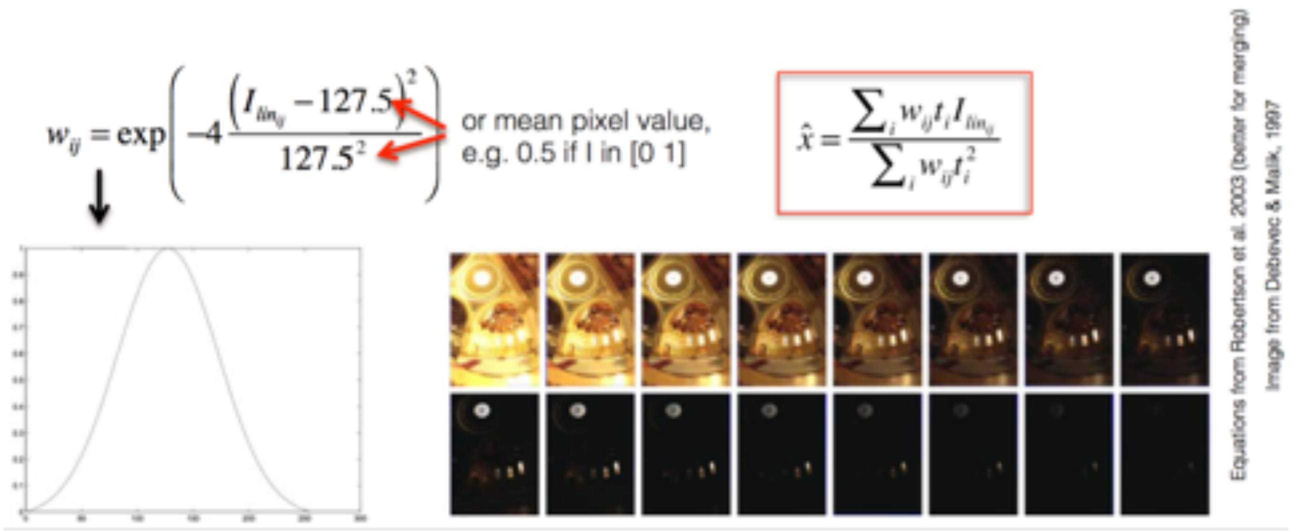
2a) HDR from linear images.

You are given a sequence of images in RAW format. Use dcraw to convert them to a linear 16 bits tiff file (dcraw -4 -T). Merge these images using the formula from Robertson's method (below). Robertson's method is actually more complicated, but for us, we will simply apply this weighting. Save the result an HDR image. Save it as an EXR image or HDR format. You are given some code to save the file in HDR format. You can try to use OpenEXR bindings in python, but might be trickier.

$$w_{ij} = \exp\left(-4\frac{(I_{lin_{ij}} - 127.5)^2}{127.5^2}\right)$$

or mean pixel value, e.g. 0.5 if I in [0 1]

$$\boxed{\hat{x} = \frac{\sum_i w_{ij} t_i I_{lin_{ij}}}{\sum_i w_{ij} t_i^2}}$$



Individual exposure is radiance (X) * exposure time (t): $I_{lin} = tX$

MORE INFOR: ROBERTSON, BORMAN, STEVENSON, "ESTIMATION-THEORETIC APPROACH TO DYNAMIC RANGE IMPROVEMENT USING MULTIPLE EXPOSURES", JOURNAL OF ELECTRONIC IMAGING 2003

2b) Simple HDR from JPEGS

Use dcraw to convert the raw images to JPEGs. The resulting images should be in sRGB space. Apply an inverse gamma curve corresponding to the sRGB to linearise them. Compute the weights using Robertson's method, and merge them as in the first task. Compare the results.

d) Try your method with the Memorial dataset.

You can get more data set to play with here: http://pages.cs.wisc.edu/~csverma/CS766_09/HDRI/DataSet/HDRImageSet.html

Task 4: Tone Mapping (40)

We will do a simple implementation of the Bilateral Filter Tone Mapping algorithm seen in class.
http://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/

You are given a simple implementation of the bilateral filter.

The basic idea is to get intensity of the image, take its log, and apply the bilateral filter. Then divide the log(intensity) by the filtered image and do some contrast correction. Then, take the inverse of the log, and multiply by the color ratios.

Here is the high-level set of operation that you need to do in order to perform contrast reduction

- `input intensity= 1/61(R20+G40+B)`
- `r=R/(input intensity), g=G/input intensity, B=B/input intensity`
- `log(base)=Bilateral(log(input intensity))`
- `log(detail)=log(input intensity)-log(base)`
- `log (output intensity)=log(base)compressionFactor+log(detail) - log_absolute_scale`
- `R output = r*10^(log(output intensity)), etc.`

You can replace the first formula by your favorite intensity expression.
You can replace the multiplication by compressionfactor by your favorite contrast-reduction curve (e.g. histogram adjustment, Reinhard et al.'s saturation function, etc.)

The memorial church result might have undergone a questionable gamma transform before our algorithm was applied. This might explain discrepancies with other implementation of our technique.

compressionfactor and log_absolute_scale are image-dependent.

compressionfactor makes sure the contrast matches a given target contrast. It is the ratio of this target and the contrast in the base layer:

```
targetContrast/(max(log(base)) - min(log(base)))
```

I use log(5) as my target, but you can use a bigger value to preserve more contrast.
log_absolute_scale essentially normalizes the image, by making sure that the biggest value in the base after compression is 1. It is equal to max(log(base))*compressionfactor
All log are in base 10.

Hints

Install **Luminance HDR** software to load and compare the results with yours. Explore the different tone mappers and their differences. You can also use this software to create, load and compare your results.

Extra 1 Camera Response Curve Calculation

Recover the camera response function and create a HDR image. We will use the method of Debevec and Malik [http://www.pauldebevec.com/Research/HDR/debevec-siggraph97.pdf]. You have some matlab code at the end of the paper for your reference. Re-implenting this should be quite **straight forward** in python.

Parameters for B are in the exif data of the images or in a supplementary file in the case of the Memorial dataset. **Plot your g function.** You should run this for each channel separately. **Use about 100 random samples for points.** And experiment with **lambda** so your function is smooth.

Extra 2 Take your own HDR Sequence!

Use the Bracketing mode for as many images as possible, typically between 3 and 9 More than 3 is better. Set the exposure to -3 +3 or similar, it will depend on your camera. If you have doubts you can bring your camera and we will check.

Deliverables

Code, images, and documentation You will demo it for marking during the next lab.