Homework 2
CSC2529: Computational Imaging 2023

Due: Oct. 4, 2023 at 11:59 pm

Learning Goals

In this homework, you will learn about the basic image formation model of a camera and explore aspects such as ray optics, depth of field, aperture, and defocus blur. Moreover, you will study and implement some of the most important parts of camera ISPs, including interpolation, demosaicking, and several variants of image denoising, such as Gaussian filtering, bilateral filtering, and non-local means.

Instructions

In this week's problem session, we will walk you through this homework step by step. It may be helpful to watch the problem session before you start working on this homework or ask questions on Piazza.

This is a programming assignment. Students are strongly encouraged to use Python for this assignment. Other programming environments may generally also be acceptable, but will not be supported in the office hours or on Piazza.

You should document all your answers, plots, and derivations in a **single pdf** file containing all requested results and the scripts that generated these results. Submit your solution to Quercus. Solutions to tasks should be on separate pages and include text, images, and code.

Task 1 of 3: Depth of Field (25 points)

Let's define the depth of field as the depth range around the focal plane of a camera that produces a circle of confusion with a diameter of 5 pixels or less. We are working with a Canon 5D Mark II digital SLR camera that has a full frame (35 mm) sensor of size 36×24 mm and a resolution of 5616×3744 pixels (21.0 MP). The camera is equipped with a Canon EF 50 mm f/1.8 lens (i.e., the focal length is 50 mm).

1. The lens is focused at a distance of 2.5 m in the scene. What is the depth of field, i.e., **near distance, far distance, and difference between far and near distance** in the scene for the following two aperture settings: f/3 and f/16? **Draw the setup.** Hint: the focal plane of 2.5 m should be in between these two distances. Comment on the difference in the depth of field between these two conditions.
2. For an aperture setting of f/3, what is the depth of field (i.e., near and far distances as well as their difference in the scene) when the lens is focused at the following two distances: 0.5 m and 20 m (in the scene)? Comment on the difference in the depth of field between these two conditions.

Hint: make sure to use the same unit for all quantities (e.g., m or mm).

Task 2 of 3: Implement a Simple ISP (35 points)

Implement a simple image processing pipeline. A linear RAW image is provided in the homework release. Plot all intermediate results.

1. Implement demosaicing using linear interpolation.

   You can use the function `scipy.interpolate.interp2d()` to interpolate each color channel.

   The example image 'lighthouse_RAW_colorcoded.png' shows you in false colors which subpixel measures which color channel. Don't use that for demosaicing, use the image 'lighthouse_RAW_noisy_sigma0.01.png'!. The demosaiced image will still be linear, apply a gamma correction to convert it to a proper sRGB image. Calculate the PSNR (after gamma correction), comparing your results to lighthouse.png.

2. Unfortunately, the lighthouse is a difficult example for demosaicing. You should see color artifacts in the fence. Go back to the linear demosaiced image, convert it to YCrCb color space and low-pass or median filter the chrominance channels but **not** the luminance channel. Convert back to RGB and then apply the sRGB gamma. Adjust the filter parameters to minimize color artifacts.

   You can use the function `skimage.color.rgb2ycbcr()` to convert an RGB image into a YCbCr image and back.

   Calculate the PSNR as in the previous section.

3. Implement demosaicing based on the paper "High quality linear interpolation for demosaicing of Bayer-patterned color images" by R. Malvar, L. He, and R. Cutler (ICASPP 2004). Apply gamma correction and report the peak signal-to-noise ratio (PSNR) between your demosaicked image and the ground truth.

   You can use the function `scipy.signal.convolve2d()` to perform the filtering/convolution with the demosaicking kernels specified in this paper.

4. Compare the interpolated results of the lighthouse of the previous sections to each other and to the original image (lighthouse.png). The comparison should be qualitative (comparing images) and quantitative (comparing the PSNR).

You can use the function `skimage.metrics.peak_signal_noise_ratio()` to compute PSNR values whenever requested here or in the following.

Task 3 of 3: Image Denoising (40 points)

Implement and compare several image denoising techniques. A noisy image and template script are provided.

1. Implement local, linear smoothing (with a Gaussian filter kernel) and local, nonlinear filtering with a median filter.

   You can use the skimage function `skimage.filters.gaussian()` and the function `scipy.ndimage.median_filter()`.

   Compare the results of at least 3 different standard deviation (σ, for linear smoothing) or different filter window sizes (for median filter) for each of these two methods (6 images total). For each of the three settings per method, try to find the approximately "best" heuristic choice of parameter (standard deviation or window size) for the example image, and choose the other two parameters to be slightly smaller and bigger, respectively. For example, if you determine $\sigma = 1$ to be the best choice, use that as one of the test cases and then use $\sigma = 0.5$ and $\sigma = 2$ as the other two.

2. Implement a bilateral filter. Test the same noisy image with the same 3 spatial filters with the same standard deviations as your choice in a) but test them for two different choices of the standard deviations for the intensity σ_i.
3. Implement the non-local means algorithm. This will be a little tricky, so here are some tips if you run into trouble:
   - When you try to find similar neighborhoods for the current window, restrict your search to some, larger neighborhood. If you try to search the entire image, it will take too long.
   - Do **not** include the window centered on the current pixel when searching for other, similar windows!
   - Rather than computing the naive L2 norm between current neighborhood and that of another pixel, use a weighted comparison. Meaning, instead of using the weight

$$w(i,j) = \frac{1}{Z(i)} \exp\left(-\frac{\left\| v\left(N_i\right) - v\left(N_j\right)\right\|^2}{h^2}\right), \forall i \neq j \tag{1}$$

     use:

$$w(i,j) = \frac{1}{Z(i)} \exp\left(-\frac{\sum_{mn}\left(k_{mn}\left(v\left(N_i\right)_{mn} - v\left(N_j\right)_{mn}\right)^2\right)}{h^2}\right) \tag{2}$$

     where k_{mn} is the weight for calculating the distance between windows N_i, N_j at location m, n inside the windows.

Bonus Task (up to 5 points extra)

Use your own camera (or one that you borrowed) and capture a RAW photograph. Extract the RAW sensor image (e.g., with the command line tool dcraw), apply demosaicing, denoising, and gamma correction on it. Compare your results with those of dcraw.

Questions?

First, review the lecture slides and videos as well as the problem session because the answer to your question is likely in there. If you don't find it there, post on Piazza, come to office hours, or email the course staff mailing list (in that order).