Homework 4

CSC2529: Computational Imaging 2022

Due: Oct. 26, 2022 at 11:59 pm

Learning Goals

In this homework, you will learn about high-dynamic-range (HDR) imaging and implement a popular method to fuse several low-dynamic-range (LDR) images into a single HDR image. You will also experiment a little bit with tonemapping algorithms. Finally, you will deepen your understanding of different types of noise in sensor images and how these affect several computational imaging techniques, including burst denoising and the flutter shutter.

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 Ed Discussion.

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 Ed Discussion.

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 Gradescope. Solutions to tasks should be on separate pages and include text, images, and code.

Task 1 of 3: HDR Image Fusion (40 points)

Implement high dynamic range image fusion. You will find 16 different low-dynamic range (LDR) exposures of the same scene (unzip the file 'hdr_data.zip') in the homework release folder. The exposure times and other relevant information are listed in the .txt file in that folder. Load these .png images. The images are provided in sRGB space, so you need to apply an inverse gamma curve to linearize them.

1. Compute the fusion weights using Debevec's method, as discussed in the lecture and problem session. Show and submit the images of your weights for each of the input LDR images as shown in the problem session.
2. Fuse all exposures into a single, floating-point HDR image. If you try to plot this directly, it won't look very good on your screen because it's an LDR display and because this HDR image encodes linear intensity values rather than the sRGB tone curve.

   To visualize your HDR image, try to find a good scale s and gamma parameter γ, which will show your normalized HDR image $I_{HDR} \in [0.0, 1.0]$ as an 8 bit LDR image as $(s \cdot I_{HDR})^{\gamma}$. Play around with these two parameters (s, γ) to make the result look as good as possible by clamping/clipping the resulting image into the range

[0.0, 1.0].

The starter code also already contains a function that will tonemap your HDR image using one of OpenCV's built-in functions, i.e., it'll compress it back to 8 bits and apply a tone curve that show all details in a way that should be much superior to any of the individual LDR images.

Report the two parameters, s and γ, that you chose for your simple gamma-based tonemapper as well as the images produced by both your gamma-based tonemapper and the tonemapping algorithm provided in the starter code.

When fusing the low-dynamic-range exposures, you need to take the logarithm of the linearized LDR images. However, some of the image values will be 0 and you want to avoid computing `log(0)` because it's `-inf` and having such values in your fused HDR image will cause trouble for other functions operating on this image. Therefore, make sure to add a small constant to your image values before taking the logarithm, i.e., use `log(values+np.finfo(numpy.float32).eps)` instead of `log(values)`.

Task 2 of 3: Burst Denoising and SNR (20 points)

In this question, we will study burst denoising in a bit more detail. The idea behind burst denoising is to take several short exposures, each being noisy, of the same scene, align them, and simply average them to compute a single denoised image. For the purpose of this homework question, we will assume that all of these images are already aligned and we want to study the signal-to-noise ratio (SNR) of the noisy and denoised images in several different conditions.

1. Assume that the only source of noise in a set of K aligned images is zero-mean Gaussian-distributed read noise with a standard deviation of σ. Each of the K images has an SNR of $\frac{\mu}{\sigma}$, where μ is the mean pixel value. What is the SNR of the **averaged** (not summed!) image expressed in terms of K, μ, and σ?
2. Assume that the only source of noise in a set of K aligned images is Poisson-distributed photon noise. Each pixel observes an average photon rate of λ, so the SNR of each of the K images is $\frac{\lambda}{\sqrt{\lambda}}$. What is the SNR of the averaged image expressed in terms of K and λ?

Task 3 of 3: Flutter Shutter and SNR (40 points)

The flutter shutter is a temporally modulated aperture pattern (either fully transparent or fully opaque at any given point in time) that modifies the recorded image within a single camera exposure. The goal is to make motion deblurring well posed. We capture images with an exposure time of 1 s and the flutter shutter can modulate the scene at 100 Hz, that is it can be open or closed for 10 ms and then instantaneously switch its state. We are looking at motions that can be perfectly deblurred up to the temporal resolution of the flutter shutter. We assume an overall 50% duty cycle, and thus across the entire (modulated) 1 s exposure, the mean pixel value μ in recorded images is $0.5 * n$, where n is the number of photons that would hit the sensor during a full (non-modulated) 1 s exposure.

I have a different idea: why not shorten the exposure time to 10 ms instead of 1 s, and take a burst of photos during the 1 s image-capture time. Assume that the delay between successive exposures is negligible (end of one exposure is the beginning of the next, no break in between). That gives me the same image sharpness, and will **double** the number of photons hitting the sensor across the full 1 s image-capture time. Sure, the Gaussian-distributed read

noise of the images will be increased, but I'll just average all 100 images, which **might** give me the same signal-to-noise ratio (SNR) for static, non-moving parts of the scene. For moving objects, I can align them in post-processing and denoise them by averaging the aligned frames. Because we are dividing up the image capture time up, the mean number of photons per exposure is $n/100$.

Consider two different cameras, but ignore possible sensor saturation:

1. A consumer camera used in normal lighting conditions ($n = 10,000$ photons per second) with a moderately high ISO setting that captures images exhibiting additive-only, zero-mean Gaussian (read) noise with a pixel value standard deviation of $\sigma = 50$ photons. Ignore other sources of noise.
2. A scientific CMOS sensor used in microscopy that is cooled and used in a photon-limited imaging application ($n = 1,000$ photons reach each sensor pixel per second); there is NO additive Gaussian noise, only Poisson-distributed photon shot noise.

Your job: derive the signal-to-noise ratio (SNR) of the reconstructed images for each of the experiments. Fill out the table below and discuss the results. Write out your calculations in detail.

	Flutter shutter SNR	Burst SNR
Consumer camera ($n = 10,000$)		
sCMOS camera ($n = 1,000$)		

Bonus Task (up to 5 points extra)

Use your own camera to capture and fuse an exposure sequence. Choose a scene which has a high dynamic range, such as a night scene with bright lights or an outdoor scene at daylight that contains very bright parts and very dim parts. You can also go to the Stanford Memorial Church and try to reproduce Debevec's exposure sequence. Use a tripod for any of these experiments to avoid movement between images. Make sure to estimate and apply the camera response function (CRF) appropriately. You can use OpenCV to estimate the CRF and fuse the LDR images for this task. This tutorial may come in handy: https://docs.opencv.org/3.4/d3/db7/tutorial_hdr_imaging.html.

Additional advice and troubleshooting for tasks 2 and 3

- Notation: σ is the standard deviation, σ^2 is the variance of a statistical distribution.
- The two important types of distributions here:
  - Gaussian or normal distribution: $\mathcal{N}\left(\mu, \sigma^2\right)$, where μ is the mean and σ^2 is the variance
  - Poisson distribution: Pois(λ), where the rate, or average number of photons, λ is both the mean (μ) and the variance (σ^2); the standard deviation is $\sigma = \sqrt{\lambda}$
- SNR: mean number of photons / standard deviation of noise (all measured either in number of photons or relative "signal" strength) = $\frac{\mu}{\sigma}$
- Signal integration:
  - the total signal when adding K signals, each of strength n, is $K \cdot n$
  - for independent Gaussian distributions: $\mathcal{N}\left(\mu_1, \sigma_1^2\right) + \mathcal{N}\left(\mu_2, \sigma_2^2\right) \sim \mathcal{N}\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$

- also, multiplying a Gaussian distribution by constant c results in $c\mathcal{N}\left(\mu, \sigma^2\right) = \mathcal{N}\left(c\mu, c^2\sigma^2\right)$
- for independent Poisson distributions: $\text{Pois}(\lambda_1) + \text{Pois}(\lambda_2) \sim \text{Pois}(\lambda_1 + \lambda_2)$

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 Ed Discussion, come to office hours, or email the course staff mailing list (in that order).