TTIC 31040: Introduction to Computer Vision
Spring 2021

Problem Set #3

Out: April 19, 2021

Due: Friday April 30, 11:59pm
Instructions: Turn in a write-up of your solution (as markdown cells) along with the code and the results as an iPython (Jupyter) notebook, including any figures or tables you generate. Please name your notebook the following:
`firstname-lastname-ps3.ipynb` and upload it on Canvas. Make sure the notebook you are submitting has been fully run, so that all the results are displayed in it.
Collaboration policy: it is OK (and encouraged) to discuss the problem set and any ideas for solving it with other students. However, in the end you must write your solution on your own. This includes writing any code and running any experiments.

1 Light direction

In this problem, we will be estimating the light source direction from an image of a ball made of specular (or "directional diffuse", for instance chrome) material, a common light calibration trick in photometry. We will assume, for simplicity, that:

1. The light source is at infinity (so the direction to it is the same for any point in the scene);
2. The image is obtained by orthographic projection, i.e., a point (X, Y, Z) in the scene is mapped to an image point with coordinates (X, Y);
3. The camera "look vector" (direction of the ray from the camera center towards the scene parallel to the optical axis) is $(0, 0, 1)$;
4. The sphere has a known radius r;
5. We have accurate measurements of the projection of the center of the sphere, (x_c, y_c), and of the specular highlight (maximally bright point on the sphere), (x_s, y_s).

(Note that in practice, we'd have to estimate x_c, y_c, x_s, y_s and perhaps r from the image.)

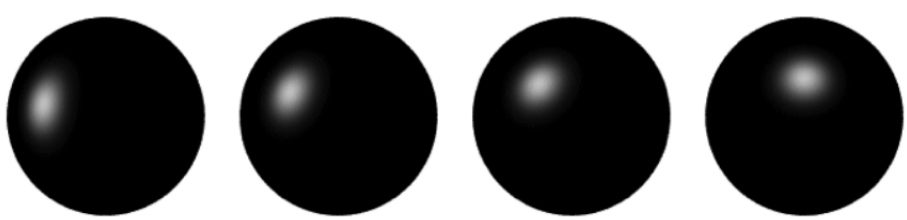

Figure 1: A directional diffuse reflection ball placed in the scene can allow us to estimate the direction of the light source. Intuitively, under some assumptions, the position of the highlight on the projection of the ball tells us where the light is relative to the camera.

Problem 1 **[15 points]**
Compute the direction of the light source (in camera coordinates) from x_c, y_c, x_s, y_s and r.

Advice: Consider the fact that points (X, Y, Z) on a sphere with center at (x_c, y_c, z_c) of radius r (including the observed specular highlight) satisfy $(X - x_c)^2 + (Y - y_c)^2 + (Z - z_c)^2 = r^2$. You may want to start by drawing a 2D scene with a 1D image (think of it as a "view from above" over you 3D scene/image plane) and work out the solution for that case, which is simpler to visualize and perhaps reason about, before deriving the full 3D solution.

End of problem 1

2 Photometric stereo

In this problem we will explore photometric "stereo". We will work with a dataset of images taken with different (known) light source directions, and estimate the albedo, surface normals, and 3D structure (shape) from the images of faces. We have provided a small subset of images from the Yale Face dataset. After unzipping the archive, you will have the dataset as `croppedyale` in your ipynb directory. Light source directions are encoded in the filenames; utility functions that extract the directions and load the images are provided.

Problem 2 **[5 points]**
For each subject (subdirectory in `croppedyale`), read in the images and light source directions. This is accomplished by the function `LoadFaceImages` which returns the images for the 64 light source directions and an ambient image (i.e., image taken with all the light sources turned off). Pick one of the subjects and plot a few of the images. Answer qualitatively: what kinds of shading cues can you see in images with different lighting directions?

End of problem 2

Problem 3 **[5 points]**
Preprocess the data: subtract the ambient image from each image in the light source stack, set any negative values to zero, rescale the resulting intensities so that each pixel $p \in [0, 1]$.

End of problem 3

Problem 4 **[20 points]**
Estimate the albedo and surface normals. For this, you need to fill in code in `photometric_stereo`, which is a function taking as input the image stack corresponding to the different light source directions and the matrix of the light source directions, and returning an albedo image and surface normal estimates. The latter should be stored in a three-dimensional matrix. That is, if your original image dimensions are h $\times$ w, the surface normal matrix should be h $\times$ w $\times$ 3, where the third dimension corresponds to the x-, y-, and z-components of the surface normals. To solve for the albedo and the normals, you will need to set up a linear system as shown in the lecture.

For each of the four subjects, display your estimated albedo maps and surface normals using `plot_albedo_and_surface_normals`.

End of problem 4

Problem 5 **[15 points]**
Compute the surface height map by integration, implementing the method described in Lecture 5. Instead of continuous integration of the partial derivatives over a path, you will simply be summing their discrete values. Your code implementing the integration should go in the `get_surface` function. There are a number

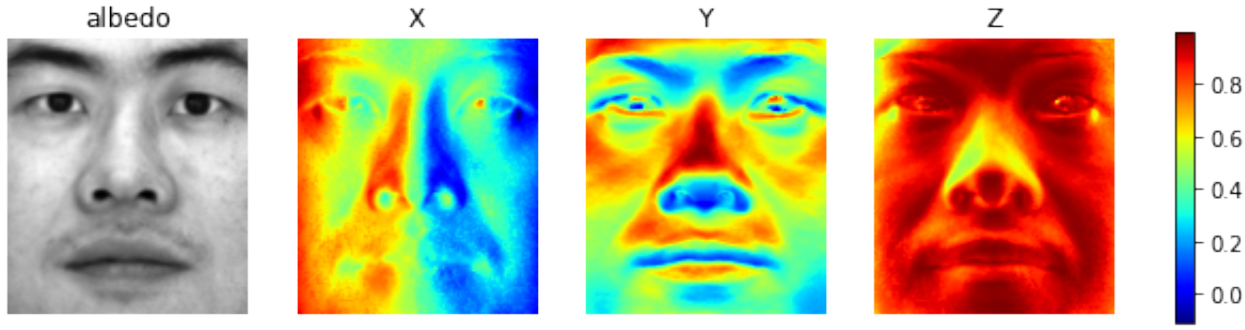

Figure 2: Solving for albedo image and surface normals. The color-mapped images show the spatial components of a unit surface normal vector estimated at each pixel. You should have similar looking results.

Figure 3: Single image direction visualization (with spherical legend) for the surface normals.

of variants for integration, e.g. integrating first the rows, then the columns: that is, your path first goes along the same row as the pixel along the top, and then goes vertically down to the pixel. You can also integrate first along the rows, then along the columns, average the two variants, or average among multiple random paths. Whichever integration scheme you pick, you should be able to recover a heightmap from your estimated normals.

End of problem 5

Problem 6 **[5 points]**
Discuss how the Yale Face dataset violates the assumptions of photometric stereo as covered in the slides. What features of the data can contribute to errors in the results? Feel free to point to specific input images to illustrate your points. Choose one subject and attempt to select a subset of all viewpoints that better match the assumptions of the method. Show your results for that subset and discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.

End of problem 6

3 Image denoising

Noise is found in nearly all digital images, and denoising (or noise reduction) is an important field in image processing. It is present as a stage in processing for many other ("higher level") computer vision tasks. Central to the denoising problem is a tradeoff between noise reduction and "over-smoothing" – it is relatively easy to eliminate all noise in most images, but the resulting procedure will also often erase all interesting detail. In this section we will explore this tradeoff, starting with the simplest (and "harshest") approaches and then exploring an interesting (but more complex) procedures.
We have provided two images: one is a clean image, which we can artificially corrupt with noise to explore

the performance of denoising algorithms relative to known "ground truth". The other is a real noisy image from the camera, in Fig. 4, for which we do not have a ground truth image and thus can evaluate denoising results only subjectively. Whenever you are asked to provide/evaluate results on images, please use these two images (with the artificial noise for the first one set to Gaussian noise with $\sigma = 0.1$).

Figure 4: Many images taken in non-ideal settings, e.g. low light or with low quality cameras, have varying degrees of noise. This image was taken by a cheap Android phone in poor light settings.

Problem 7 **[5 points]**
Implement a denoising function using simple mean filter (using `np.convolve` or your favorite Python convolution function, and the appropriate kernel), experimenting with different kernel sizes. What do you observe about the tradeoff between kernel size and noise? Does this filter do a good job of preserving detail/removing noise?

End of problem 7

Problem 8 **[5 points]**
Implement a denoising function using median filter (the same filter used for Pset1). What do you observe about the tradeoff between kernel size and noise? Does this filter do a good job of preserving detail?

End of problem 8

Problem 9 **[5 points]**
There are a variety of quantitative metrics that have been proposed to evaluate denoising techniques, most of them known to be an imperfect match to perceptual quality judgments. The gold standard for image quality evaluation (an inherently subjective task) is still the discerning human eye. For the purposes of this problem set, we will use two simple metrics – the mean square error (MSE), and a signal-processing derived metric known as the peak signal-to-noise ratio (PSNR). According to Wikipedia, PSNR is "an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation." It is defined as follows, given the max possible pixel value for the image (usually 255 but sometimes a higher bit depth):

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{1}$$

Implement a function that computes the MSE and PSNR between a noisy image and a denoised image, and quantitatively evaluate the median filtered and mean filtered images using these metrics. How do they compare? Do the numbers correspond to your own subjective notion of "quality"?

End of problem 9

Problem 10 **[15 points]**

Nonlocal means. [1] As described in the lecture, non-local means is a denoising filter that averages values of pixels centered on similar patches. Complete the implementation of `nlmeans_filter`, and compare its output on the two images to the previous two filters you implemented, but quantitatively and qualitatively. Did this filter do a better job of preserving detail?

End of problem 10

Problem 11 **[15 points]**

Bilateral filtering. [2] Implement the bilateral filter for denoising, and compare its output on the two images to the previous two filters you implemented, but quantitatively and qualitatively. Did this filter do a better job of preserving detail?

End of problem 11

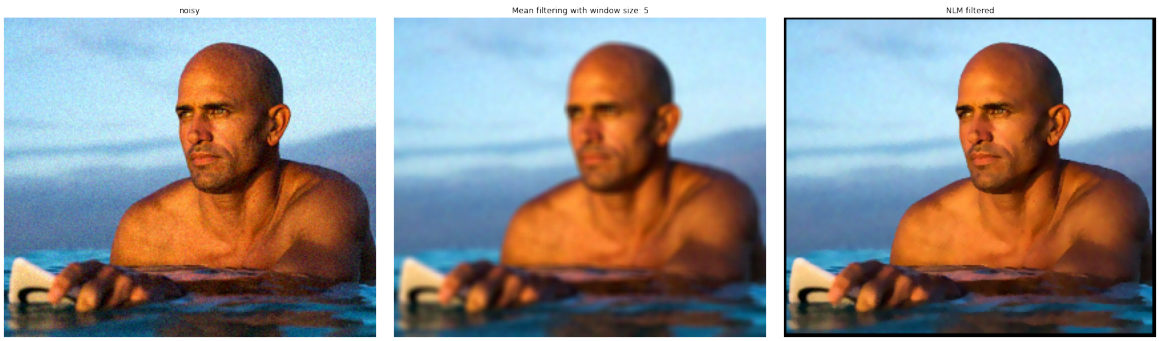

Figure 5: Mean filtering (middle) leads to much more detail loss than NLM.

[1] Note that given your implementation, this filter could take several minutes to run.

[2] The points in this homework add to 110, but it will still be graded out of 100. You are still encouraged to implement both the bilateral and non-local means filters.